The Thesis Committee for Joel David Marquez Jr.
Certifies that this is the approved version of the following thesis:

Elastic Network & Finite Element Model to Study Actin Protein Mechanics & Its Molecular Elasticity

APPROVED BY
SUPERVISING COMMITTEE:

Supervisor:

Tess J. Moon, Supervisor

Pengyu Ren

Elastic Network & Finite Element Model to Study Actin Protein Mechanics & Its Molecular Elasticity

by

Joel David Marquez Jr., B.S.M.E.

Thesis

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Engineering

The University of Texas at Austin

December 2010

I would like to dedicate this work to my family. You all mean the world to me.

Acknowledgements

I would like to thank my advisor Tess Moon for her guidance and advice throughout my tenure as a Master's student. Her mentoring and direction has helped to shape this work to what it is today. I would also like to thank my lab mates Dennis Liu, Esfandiar Khatiblou and Steven Kreuzer for their generous support and feedback throughout my work. Their presence in the lab has helped to foster a healthy and fun working environment. I also thank my friends for being there for me and always putting a smile on my face.

And of course, I would also like to thank my family for always being there for me. The last few years of my life have been quite a journey, and I thank you for your love and patience while have I completed this work.

December 3, 2010

Abstract

Elastic Network & Finite Element Model to Study Actin Protein Mechanics & Its Molecular Elasticity

Joel David Marquez Jr., M.S.E.

The University of Texas at Austin, 2010

Supervisor: Tess J. Moon

While there have been many recently developed Elastic Network Models (ENM) to calculate the fluctuation dynamics of proteins, e.g., Gaussian Network Model (GNM), Anisotropic Network Model (ANM), Distance Network Model (DNM), the concept of loading these models to study the molecular mechanics and constitutive behavior of structural proteins has remained relatively untouched, until very recently. This work entails using the ANM as the framework for developing a finite element model of a 9–monomer strand of actin. Critical input parameters to the model, such as the cutoff radius, r_c, and spring constant, k, are generated by matching the all-atom steered molecular dynamics (SMD) residue displacements to that of the ANM. The parameters yielding the best match between the SMD and structural ENM (SENM) simulations will then be input into the finite element model (FEM) for a more in depth analysis.

The finite element model incorporates a 9–monomer strand of actin. The F–actin strand is subjected axial and torsional loads comparable to those seen *in vivo*. Key areas of interest in the protein are examined, such as the nucleotide binding pocket (NBP) and the DNase I binding loop, to demonstrate how loading affects the protein's conformation. Local residue displacements are tracked in an effort to garner a better understanding of how various loads are transmitted through F–actin during key events. Insights and conclusions are discussed along with the implications of this work.

Table of Contents

Acknowledgments ..... **v**

Abstract ..... **vi**

List of Tables ..... **xi**

List of Figures ..... **xii**

Chapter 1 Introduction **1**

1.1 VARIOUS ROLES ACTOMYOSIN PLAYS IN CELLS 1

1.2 COMPUTATIONAL LIMITATIONS 3

1.3 QUESTIONS THIS WORK WILL ANSWER 4

1.3.1 Unloaded vs. Loaded Network Parameterization 5

1.3.2 Key residues that withstand large inter-monomer load transer 6

1.3.3 Enhanced understanding of nucleotide binding pocket 6

Chapter 2 Biology and Theory Background **7**

2.1 BIOLOGY 7

2.1.1 Globular and Filamentous Actin Structure 7

2.2 THEORY 11

2.2.1 Gaussian/Anisotropic Network Models 12

2.2.2 Debye-Waller (B) Factors 17

2.2.3 Modifications to ENM/FEM theory 18

2.2.4 Calculix 19

2.2.5 MD/SMD 20

2.3 EXPERIMENTAL DATA 21

2.3.1 Actin constitutive properties 21

2.3.1.1 *Axial Stiffness* 22

2.3.1.2 *Torsional Stiffness* 22

2.3.2 Residue Specific pulling experiments 23

2.4 EXISTING ACTIN MODELS 24
2.4.1 MD simulations and course graining 24
2.4.2 FEM models 26
2.4.3 Additional Network Models 27
2.5 SUMMARY 30

Chapter 3 Model System and Methodology 32
3.1 MODEL SYSTEM 32
3.1.1 9mer F-actin structure 33
3.2 METHODOLOGY 34
3.2.1 Unloaded 9mer ENM Parameterization 34
3.2.2 Decoupling r_c and k 36
3.2.3 Axial Loading 38
3.2.4 Torsional Loading 40
3.2.5 Residual/Model Validation 42
3.3 QUANTITIES OF INTEREST 45
3.3.1 B1 and B2 45
3.3.2 The Dihedral Angle 46
3.3.3 Intermer Distance 47
3.4 SUMMARY 48

Chapter 4 Model Parameterization and Loading Results 50
4.1 INTRODUCTION 50
4.2 UNLOADED MODEL PARAMETERIZATION 50
4.3 AXIAL LOADING RESULTS 52
4.3.1 SMD Axial Results 52
4.3.2 Axial Load Parameterization/Validation 54
4.3.3 SENM Axial Results 63
4.4 TORSIONAL LOADING RESULTS 67
4.4.1 SMD Torsional Results 68
4.4.2 Torsional Load Parameterization/Validation 70

4.4.3 SENM Torsional Results 72
4.4.4 QoI summary table 74
4.5 UNLOADED VERSUS LOADED PARAMETERIZATION COMPARISON 78
4.6 DISCUSSION 79
4.7 SUMMARY 83
Chapter 5 Conclusions and Future Work 84
5.1 EFFECTIVENESS OF THE SENM MODEL 84
5.2 FUTURE WORK 85
Appendix A FORTRAN 90 Normal Mode Analysi Code Modifications 88
Appendix B Calculation for F-actin Extension under Axial Load 107
Appendix C Calculation for F-actin Twist under Torsional Load 109
Appendix D FORTRAN 90 Brute Force Residual Calculation Code 111
Appendix E *CalculiX* Axial Loading Input File Code 119
Appendix F *CalculiX* Torsional Loading Input File Code 122
Bibliography 125
Vita 130

List of Tables

Table 1. ENM Comparison between various models.....28

Table 2. Global and local numbering of residues under torsion load. 41

Table 3. Summary of SENM model parameterization under axial loading 63

Table 4. Summary of SENM model parameterization under torsional loading 70

Table 5. QoI table summarizing the results of the SMD and SENM simulations 74

List of Figures

Figure 1. Eukaryotic cell with various cytoskeletal components labeled. 1

Figure 2. Various mechanisms of mechanical force transduction 2

Figure 3. Ribbon model and cartoon drawing of G-actin structure. 8

Figure 4. View of critical G-actin loops and close up of NBP 9

Figure 5. 9mer F-actin structure to be used in the model 9

Figure 6. G-actin binding in the filament state. 10

Figure 7. The actin treadmilling process 11

Figure 8. Overview of the ENM representation 13

Figure 9. Simple 1D spring system with 2 nodes and forces applied at each node 18

Figure 10. Actin length versus tension curve for a 19 μm long filament 23

Figure 11. CG procedure employed by Chu and Voth 25

Figure 12. 9mer F-actin structure to be used in the model. 32

Figure 13. Flowchart showing the methodology to determine optimum r_c and k 35

Figure 14. Image of the r_c versus k plane space to be scanned 37

Figure 15. 9mer F-actin structure subjected to an approximated axial load. 39

Figure 16. 9mer F-actin torsional loading locations 40

Figure 17. View looking down the filament axis 42

Figure 18. Close up of the NBP showing the distances B1 and B2 46

Figure 19. Image of a G-actin monomer, and its relation to the dihedral angle, φ_d. 47

Figure 20. The QoI showing the distance between mer5/res 288 and mer7/res 288 48

Figure 21. Unloaded r_c versus k and PCC for the 9mer structure 51

Figure 22. Mer5 residue displacements from a 200 pN axial load 53

Figure 23. Ribbon representation of G-actin showing SMD residue displacements 54

Figure 24. Axial loading r_c versus *k* plane for the $\Delta L_{19}/L_{19}$ metric..... 55

Figure 25. Axial loading r_c versus *k* plane for the inverse, or $L_{19}/\Delta L_{19}$ metric. 56

Figure 26. SENM Dali score surface plot of r_c versus *k* under axial loading 57

Figure 27. Axial loading r_c versus *k* plane for the $\Delta B_1/B_1$ metric 59

Figure 28. Axial loading r_c versus *k* plane for the $\Delta B_2/B_2$ metric 60

Figure 29. Axial loading r_c versus *k* plane for the $\Delta D_{inter}/D_{inter}$ metric 61

Figure 30. Axial loading r_c versus *k* plane for the $\Delta\varphi_d/\varphi_d$ metric..... 62

Figure 31. Screen shot of the entire axially loaded 9mer structure from GraphiX. 64

Figure 32. Overlay of Mer5 SMD and SENM structures..... 65

Figure 33. Mer5 SENM residue displacements for a 200 pN axially applied load. 66

Figure 34. Residue displacements from the SENM model under axial load. 67

Figure 35. Mer5 SMD residue displacements under torsional loading. 68

Figure 36. Mer5 SMD displacements under torsional loading – ribbon structure..... 69

Figure 37. SENM Dali score surface plot of r_c versus *k* under torsional loading. 71

Figure 38. Mer5 SENM residue displacements for the 200 pN-nm torque load. 72

Figure 39. SENM model residue displacements under torsional loading 73

Figure 40. Side view of SMD mer5 under torsional load..... 78

Chapter 1

Introduction

1.1 VARIOUS ROLES ACTOMYOSIN PLAYS IN CELLS

The actomyosin complex, and more specifically the actin filament (F-actin), in the context of this thesis, plays a very large role in both muscle and non muscle cells in the body [1]. Structurally, F-actin is arranged in multiple configurations, depending on the size of applied loads and its location within a cell.

Functionally, F-actin acts as the main load bearing member in many cellular processes involving the motor protein myosin, including: **spreading**, where small portions, or protrusions, of a cell extend into neighboring regions, as seen in filopodia

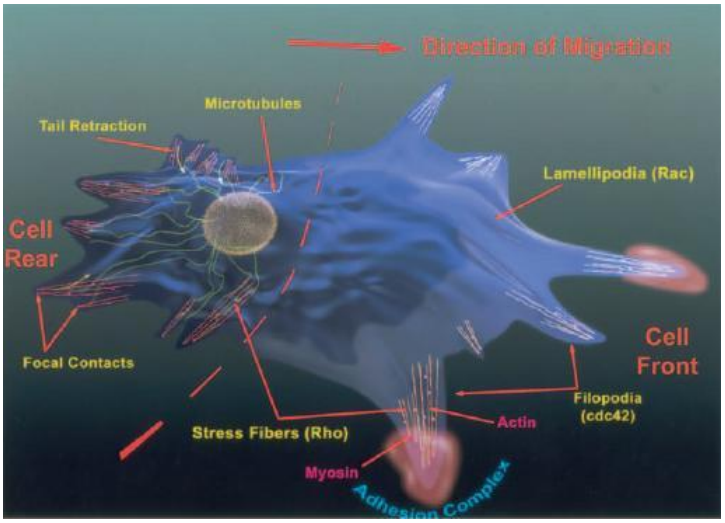


Figure 1. Eukaryotic cell with various cytoskeletal components labeled. In addition, various structures involving actin are shown, including filopodia, lamellipodia, focal contacts and stress fibers [2].

and lamellopodia (Figure 1) [3, 4]; **migration**, where a cell will break away from its local environment and move through tissue[5]; **adhesion**-the binding of a cell to the extracellular matrix (ECM) or another cell (Figure 1) [6, 7]; **cytokinesis**-the process of a cell's cytoplasm splitting in two to form two daughter cells via mitosis or meiosis[8]; and **differentiation**, where an unspecialized cell will adapt to its local environment and become more tissue specific[9, 10].

In addition to these coarse, whole cell processes, F-actin also plays a very active localized role in intracellular signal transmission, based on the size of the applied forces and the specific protein that actin is bound to [11]. According to Dr. Wang of the University of Illinois at Urbana-Champaign, the speed of mechanical signals is second only to the electrical impulses that are transmitted down neurons in biological signaling

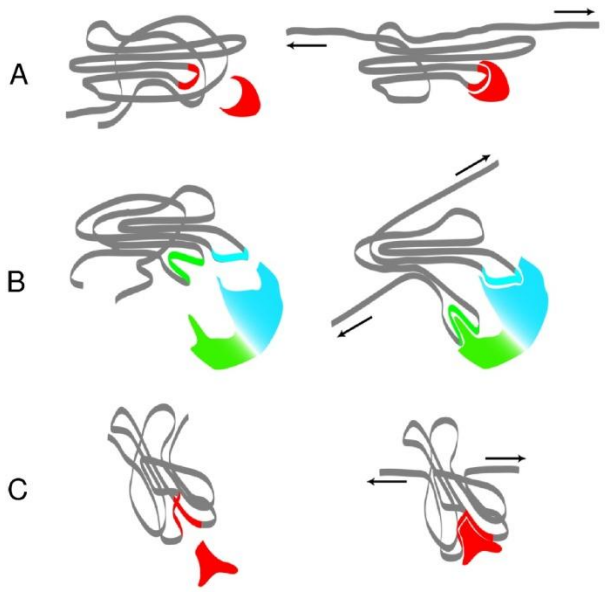


Figure 2. Various mechanisms of mechanical force transduction into a biochemical signal. A) External forces deform protein to expose a cryptic binding site. B) Distance between binding sites is altered by external load. C) Binding site itself is deformed to allow for ligand binding [11].

[12]. By mechanisms such as exposing cryptic binding sites, changing the distance of binding sites, and changing the shape of a binding site, as seen in Figure 2, mechanical forces are converted into biochemical signals [11, 13, 14]. Therefore, quantifying how load is transmitted through F-actin will allow us to have a more detailed understanding of how the F-actin structure rearranges itself under load and how it imparts that load to its neighboring focal adhesion signaling proteins.

1.2 COMPUTATIONAL LIMITATIONS

In light of the recent discoveries of the importance of mechanical force transmission through proteins, one of the major goals of this work is to assess the effectiveness of a more computationally efficient method to track residue level movements (specifically displacements) during the application of loads to proteins. While the model system to be investigated, F-actin, is not known to undergo any large conformational changes while in its filamentous state (however, the globular actin protein, G-actin, does), F-actin proves a great model system to use because it has been extensively studied in experiments, and numerous constitutive mechanical properties about the system are known [15-20].

While many molecular dynamic (MD) simulations on G and F-actin have been performed under non-loaded conditions, including the nucleotide effects on the actin structure [21-23], the author has not been able to find any sources using MD simulations to investigate the effect of load on F-actin (also called steered molecular dynamics (SMD) due to the 'steering' of load application). This is most likely due to the large computational cost incurred in running such a large system. In addition to the complex potentials that characterize intra-molecular interactions, MD simulations also have a large

computational burden because of the inclusion of water molecules (which alone accounts for 90 and 95% of the atoms in a system [24]). The higher fidelity of MD simulations can increase the number of degrees of freedom (dof) from and elastic network model (ENM) by 100 fold. For example, running an SMD simulation on a 9 monomer structure of F-actin with 96 cores requires over 4 hours for a 500 ps simulation, while applying the same load to the structure in the *CalculiX* FEM package requires around 2 seconds on a single quad core computer. Therefore, developing an ENM based coarse graining (CG) technique to study the displacements of a protein's residues under load allows for a larger number of loading scenarios to be studied because the ENM simulations can be performed as a single deterministic run while SMD has to iterate through time.

1.3 QUESTIONS THIS WORK WILL ANSWER

There are currently no known structural protein models that have quantified the deformation behavior of the F-actin protein at the residue level. The closest known study that satisfactorily quantified an ENM's prediction to experimental pulling results investigated proteins such as green fluorescent protein (GFP), ubiquitin (Ub) and E2lip3 systems other than F-actin [25]. Therefore, one of the main goals of this work is to uncover the basic mechanisms of load transmission in the F-actin system by focusing specifically on the deformations and displacements of a G-actin protein embedded in an actin filament.

However, while we have presented the motivation to study the F-actin protein system, there is currently no known experimental data that quantifies the structural response of F-actin under load at the residue level. Because of the issue with validating

the accuracy of the structural ENM (SENM) technique, SMD simulations that mimic the loading conditions applied to the SENM will be run in parallel. Although the SMD results are from a computational model itself (as opposed to experimental data), they are becoming widely accepted among the scientific community for its ability to make accurate, atomistic level predictions of proteins under a wide array of conditions. Therefore, running SMD simulations will allow us to make a side-by-side comparison to the results generated from the proposed SENM model.

As with any type of experimental results, there is always an error associated with the desired quantity of interest, and the results from the SMD simulations are no different. Therefore, this work will present the results of the SMD simulations along with the associated error and standard deviations corresponding to each predicted term. That way, the results from the SENM simulations can be compared to those of the SMD simulations and a more quantitative assessment of the SENM's predictive capabilities can be made based on the results between the two computer models.

1.3.1 Unloaded versus Loaded Network Parameterization

The ENM has been used extensively in predicting protein fluctuation dynamics, and is generally parameterized from one independent model parameter, the cutoff radius, r_c, in order to determine the model's spring constant, k. However, since the model is going to be applied to a different type of loading scenario, it is not known if these two network parameters can still be coupled together. Therefore, as will be discussed further in section 3.2.2, the SENM will be parameterized based on treating r_c and k as two independent parameters.

1.3.2 Key residues that withstand large inter-monomer load transfer

In a bound filament state, G-actin is known to bind to four of its neighboring monomers [1]. In spite of this information, it is currently unknown how these forces are transmitted between these binding sites. This model will be able to quantitatively elucidate which residues carry the greatest/smallest loads and how these loads are distributed to their neighbors. This information will prove very useful in further understanding of how the F-actin protein system, and, more broadly, load bearing proteins distribute their loads. Based on numerous predetermined quantities of interest (QoIs) that will be discussed further in section 3.3, the state of mer5 in each loaded SENM scenario can be better described based on these specific values.

1.3.3 Enhanced understanding of the nucleotide binding pocket (NBP)

While in the filamentous state, actin will always have a nucleotide bound to it, whether it is adenosine triphosphate (ATP) or diphosphate (ADP) [26]. However, actin can be found with no nucleotide when bound to the actin related protein (ARP) profilin, which is involved in the nucleotide exchange process for G-actin [27]. When the NBP is void of any nucleotide, the binding cleft is said to be in the open position; otherwise, the NBP binding cleft is closed [28]. Because the NBP plays such a key role in the structure of actin, this work will further investigate the positions and displacements of residues in this region under various loading conditions.

Chapter 2

Biology and Theory Background

2.1 BIOLOGY

While the discussion of the biology of actin and its extensive involvement within a eukaryotic cell could be a book in and of itself, the purpose of this section is to briefly, yet sufficiently, introduce actin within the context of this work. The specific model system of both G-actin and F-actin will be discussed, along with some of the dynamic properties of F-actin.

2.1.1 Globular and Filamentous Actin Structure

On the Protein Data Bank's (PDB's) website, there are dozens of G-actin structures that can be found in various nucleotide states (ADP, ATP, no nucleotide). However, many of these structures are bound to other proteins and ligands that help stabilize the monomeric G-actin structure, such as profilin [29], gelsolin [30] and the DNase I complex [31]. Recently though, Oda et al. determined the structure of a G-actin molecule while in its filamentous conformation (pdb code: 2zwh) [32]. In collaboration with colleague Jun Zhou, a 9 monomer (9mer) segment of the actin structure was generated using a transformation procedure that superimposes the G-actin monomers based on their binding characteristics. The resultant structure is used as the starting point for the SMD and SENM simulations

The building block of F-actin, G-actin, is a globular 42 kDa protein with 375 residues. The structure of G-actin can be broken down into 4 smaller subdomains (SDs) (Figure 3.a), where SD1(red) spans residues 1-32, 70-144 and 338-375, SD2(blue) spans

residues 33-69, SD3(green) spans residues 145-180 and 270-337 and SD4(yellow) spans residues 181-269. Figure 3.b shows the N and C termini of the protein in SD1, along with the residue numbering. The protein has an asymmetric morphology around the NBP with two distinct ends; one being the barbed, or plus (+) end, and the other, which is where the the binding cleft opens, is called the pointed, or minus (-) end.

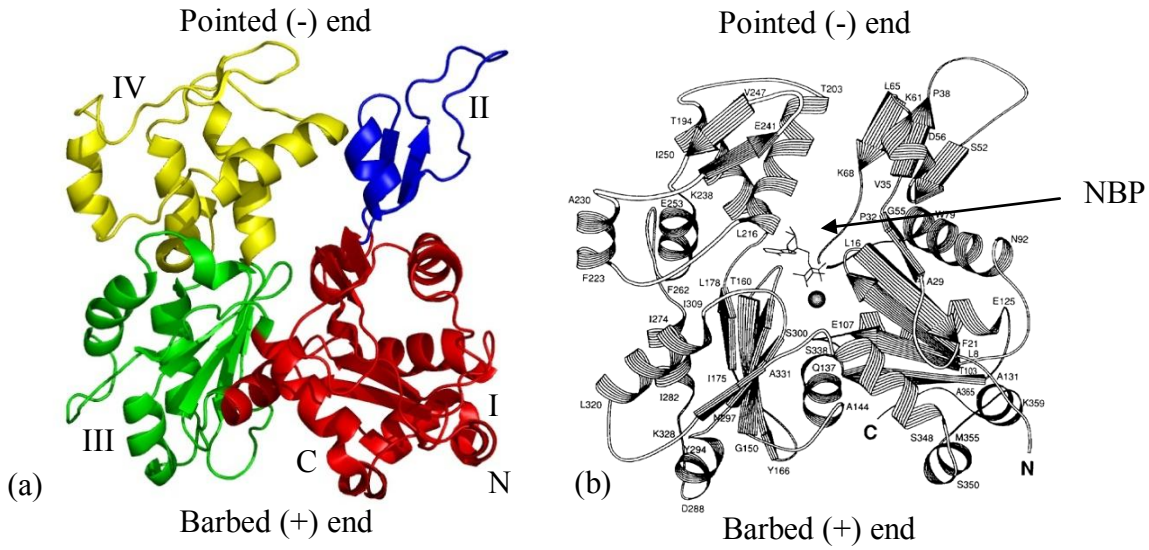


Figure 3. Ribbon model and cartoon drawing of G-actin structure. (a) shows labeled subdomains colored according to region (figure generated with the molecular graphics system PyMOL [33] , and (b) shows nucleotide (ATP/ADP) in the NBP, along with residue numbers [34] (PDB structure: 2zwh).

As seen in Figure 4, the nucleotide binds in the center of the protein, stabilized by hydrogen bonding between both the G-loop (magenta) and S-loop (blue) [23]. Around residues 41-48, highlighted in red, the DNase I binding loop, known to play a key role in stabilizing the F-actin structure, undergoes a conformational change when ATP is hydrolyzed and changes morphology from a disordered loop to an α-helix [23, 31, 35]. The F-actin-ATP structure is thought to be more stable because the larger, more

disordered, loop found with the ATP nucleotide binds to more regions of the adjacent monomer than with the smaller, more organized α-helix found when bound to ADP.

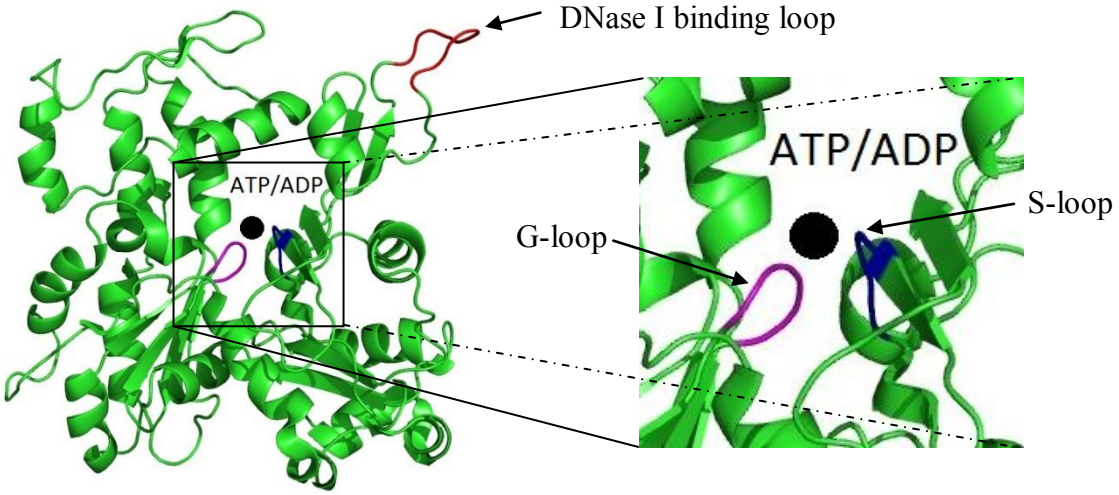


Figure 4. View of critical G-actin loops and close up of NBP. The g-loop and s-loop help to stabilize the nucleotide via hydrogen bonding and the DNase I binding loop, shown in red (DB-loop), helps to stabilize actin in the filamentous state (figure generated in PyMOL).

G-actin bound to ATP polymerizes into a filament with the help of the sequestering protein profilin, with growth occuring faster at the barbed (+) end of a

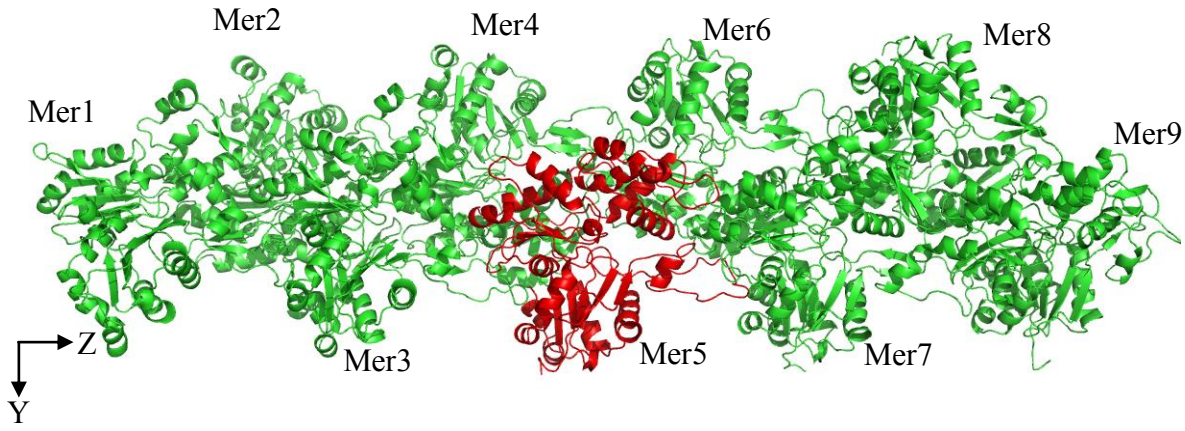


Figure 5. Ribbon representation of the 9mer F-actin structure to be used in the model. The middle monomer, mer5, is highlighted in red (figure generated in PyMOL).

developing filament. Upon binding, G-actin undergoes a conformational change where SD1 and 2 rotate relative to SD3 and 4, resulting in a slightly different energy landscape for the NBP. This new G-actin conformation encourages ATP hydrolysis, which occurs shortly after a monomer is stabilized in a bound filament state.

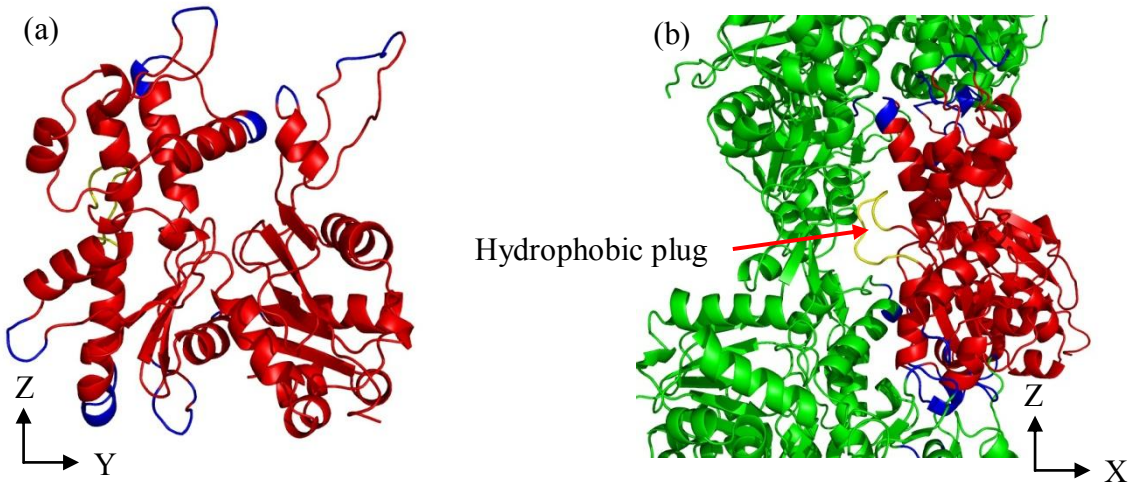


Figure 6. (a) In the filament state, a single G-actin monomer binds extensively to its neighbors. Regions that bind to neighboring monomers are highlighted in blue. (b) Shown in the bound filament state, the hydrophobic plug, highlighted in yellow, helps to stabilize the filament structure by occupying the void space within the filament (figures generated in PyMOL).

Figure 5 shows the orientation of a G-actin monomer embedded in a 9mer strand of F-actin. As seen in the figure, an F-actin monomer binds to four of its neighbors, helping to stabilize each monomer in its bound state. According to Lorenz et al., for any given monomer, residues 41-45 are known to bind to 166-169 and 375 of the adjacent monomer, 63-64 to 166, 169, 171, 173, 285 and 289, 110-112 to 195-197, 202-204 to 286-289 and 243-245 to 322-325 (Figure 6 (a)). In addition, the loop from residues 264-273, known as the hydrophobic plug, is known to help seal the inter-filament space from the surrounding water molecules (Figure 6 (b)) [34].

F-actin behaves as a stiff, helical polymer with a diameter around 7-9 nm, a persistence length of ~ 17 μm , and is , on average, found with lengths ≤ 1 μm *in vivo* [15, 36]. Morphologically, F-actin appears as two, right handed helices with a pitch of 36 nm that repeat every 13 monomers.

In the treadmilling process (Figure 7), actin filaments appear to be moving across the cytosol. Depending on the intracellular concentration of G-actin, G-actin-ATP will bind at the (+) end of a growing filament, and G-actin-ADP will dissociate from the (-) end in a steady state manner. It is this process that is responsible for cells having the ability to spread and migrate to other areas of the body (as previously mentioned in Chapter 1).

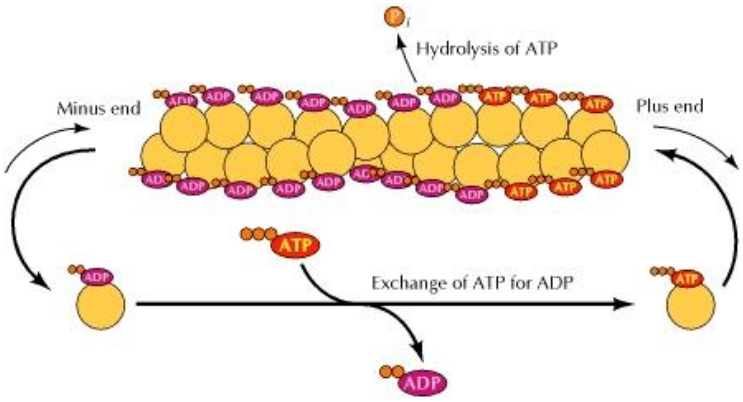


Figure 7. The actin treadmilling process [37]. G-actin-ATP preferentially binds to the plus end of a growing filament, and subsequently hydrolyzes ATP. As the filament grows and ages, the weaker G-actin-ADP structure causes G-actin-ADP to dissociate from the minus end. G-actin-ADP then binds to the protein profilin (not shown) to exchange ADP for ATP.

2.2 THEORY

Based on the original hypothesis that an ENM can be used as a framework to determine the structural response of a protein under load, a bulk of the theory to be

presented has roots stemming from the field of modal analysis. It has been widely accepted that the dynamic response of an unloaded protein, due to frequency excitation, corresponds to functionally significant conformational changes within that protein [38]. In addition, the following mentioned Gaussian Network Model (GNM), and more specifically the Anisotropic Network Model (ANM), provide for very simple ENM implementations to systematically represent a protein in an FEM framework.

2.2.1 Gaussian/Anisotropic Network Model

The ANM was developed as a modification to the GNM as a way to take into account the anisotropic, or directional, behavior of protein vibrations [39]. Originally though, the GNM was an early attempt to develop a way to model the dynamic motions of proteins, where the fluctuations were assumed to be Gaussian and isotropic in nature [40]. This type of model allows for a normal mode analysis (NMA) of the protein, which is important because the lowest modes of proteins are shown to mimic their biologically important conformational, and hence, functional changes [38, 41]. This model is also attractive because of its simplicity, relying on only a single parameter, r_c, as an input to the model [42].

The GNM seeks to simplify protein structure to the residue level where each residue is approximated by its α-carbon, C_α, location (Figure 8, (b)) [40]. The C_α's are treated as mass-less units, and are connected to each other with linear, Hookean springs. The residues are connected to neighbors based on a predefined cutoff radius, r_c, whose uniform spring constant, *k*, is determined by a fit to (MD) data (discussed later).

Beginning with the notion that the model is a collection of springs that follow Hooke's Law, and using the notation described in Figure 8(a), the potential, in its simplest form, derived from the vibration of nodes i and j is related via

$$V = \frac{1}{2}k(s_{ij} - s_{ij}^0)^2 \tag{2.1}$$

where s_{ij}^0 and s_{ij} represent the equilibrium and instantaneous distances between the two residues, respectively. Also seen in Figure 8 (a) are the equilibrium locations of residues i and j, represented by $\mathbf{R_i^0}$ and $\mathbf{R_j^0}$.

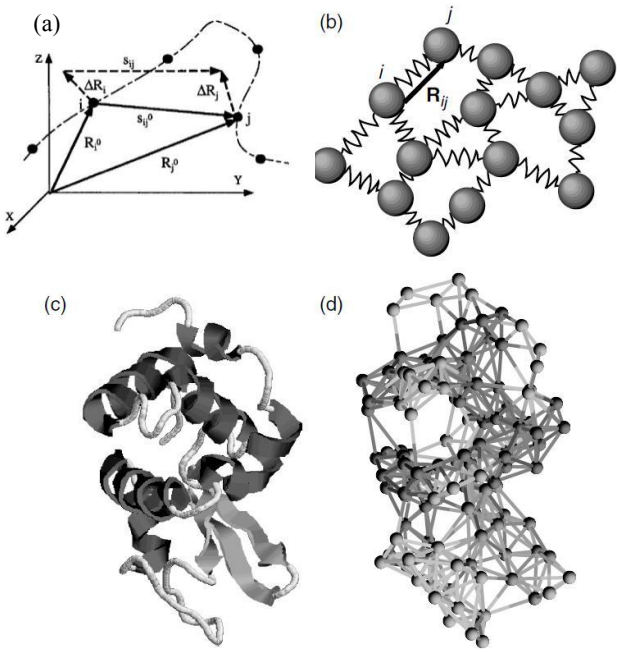


Figure 8. Overview of the ENM representation. (a) Individual residue positions and fluctuations are recorded using, for example on residue **i**, the $\mathbf{R^0_i}$ and $\Delta\mathbf{R_i}$ notation. (b) Protein structure is simplified by replacing a residue with a point located at its C_α. Each residue is connected to all of its neighbors within a specified cutoff radius, r_c, by a spring with stiffness *k*. Implementation of the ENM transforms a protein represented by its ribbon structure in (c) to the topology shown in (d). [43]

In order to solve for the frequencies and modes of vibrations of the system, inter-residue potentials are aggregated into a 'force constant matrix' called the Hessian, $\boldsymbol{H}$, where for N residues, a 3N X 3N matrix develops

$$\boldsymbol{H} = \begin{pmatrix} \boldsymbol{H}_{11} & \cdots & \boldsymbol{H}_{1N} \\ \vdots & \ddots & \vdots \\ \boldsymbol{H}_{N1} & \cdots & \boldsymbol{H}_{NN} \end{pmatrix} \tag{2.2}$$

The Hessian matrix is actually composed of N X N super elements, $\boldsymbol{H}_{ij}$, of size 3 X 3 where each super element matrix represents the interaction between element i and j. Each super element, $\boldsymbol{H}_{ij}$, is calculated by the second derivative of the potential

$$\boldsymbol{H}_{ij} = \begin{bmatrix} \frac{\partial^2 V}{\partial X_i \partial X_j} & \frac{\partial^2 V}{\partial X_i \partial Y_j} & \frac{\partial^2 V}{\partial X_i \partial Z_j} \\ \frac{\partial^2 V}{\partial Y_i \partial X_j} & \frac{\partial^2 V}{\partial Y_i \partial Y_j} & \frac{\partial^2 V}{\partial Y_i \partial Z_j} \\ \frac{\partial^2 V}{\partial Z_i \partial X_j} & \frac{\partial^2 V}{\partial Z_i \partial Y_j} & \frac{\partial^2 V}{\partial Z_i \partial Z_j} \end{bmatrix} \tag{2.3}$$

For example, row 1 column 2 of an off-diagonal term in a super element would take the form

$$\frac{\partial^2 V}{\partial X_i \partial Y_j} = -k \frac{(X_j - X_i)(Y_j - Y_i)}{s_{ij}^2}. \tag{2.4}$$

For the diagonal super elements, $\boldsymbol{H}_{ii}$, those entries are calculated by

$$\boldsymbol{H}_{ii} = -\sum_{j|j\neq 1} \boldsymbol{H}_{ij} \tag{2.5}$$

where again, for example, $\boldsymbol{H_{11}}$, would be

$$\boldsymbol{H}_{11} = k\sum_{j}(X_j - X_1)^2/{s_{1j}}^2 \tag{2.6}$$

In every super element of $\boldsymbol{H}$, k is a constant that appears at the front of every entry. This allows k to be factored out of from $\boldsymbol{H}$, which greatly simplifies the calculation of the spring constant for the system.

The topology of the protein connections can be summarized in the Kirchoff interaction matrix, which takes the form

$$\Gamma_{ij} = \begin{cases} -1 & s_{ij} \leq r_c \\ 0 & s_{ij} > r_c \end{cases} \tag{2.7}$$

and

$$\Gamma_{ii} = -\sum_{k,k\neq 1}^{N} \Gamma_{ik} \tag{2.8}$$

Incorporating the Kirchoff matrix into the Hessian matrix gives it the form

$$\boldsymbol{H}_{ij} = \frac{k\Gamma_{ij}}{(s_{ij}^0)^2}\begin{bmatrix} X_{ij}X_{ij} & X_{ij}Y_{ij} & X_{ij}Z_{ij} \\ Y_{ij}X_{ij} & Y_{ij}Y_{ij} & Y_{ij}Z_{ij} \\ Z_{ij}X_{ij} & Z_{ij}Y_{ij} & Z_{ij}Z_{ij} \end{bmatrix} \tag{2.9}$$

Where $X_{ij}, Y_{ij} and\ Z_{ij}$ represent the components of the s_{ij}^0 vector.

The normal mode frequencies, λ, and mode shapes, $\boldsymbol{u_i}$, can be determined by decomposing the inverse of the Hessian matrix into a combination of λ and $\boldsymbol{u_i}$ where

$$\boldsymbol{H}^{-1} = \sum_{i=1}^{3N-6} \frac{1}{\lambda_i} \boldsymbol{u}_i \boldsymbol{u}_i^{\mathrm{T}} \tag{2.10}$$

Note that the summation only goes to 3N-6 because the first 6 eigenvalues (and associated eigenvectors) correspond to rigid body translations and rotations, and do not contain any modal information.

After solving for the inverse hessian, there are a few key parameters that we can calculate to learn more about a protein's dynamic response. The cross-correlation of the equilibrium fluctuations between residues can be determined from $\boldsymbol{H}$, where, for residues *i* and *j*, we have

$$\langle \Delta \boldsymbol{R}_i \Delta \boldsymbol{R}_j \rangle = \frac{k_{\mathrm{B}}\mathrm{T}}{k} \mathrm{tr}(\boldsymbol{H}_{ij}^{-1}). \tag{2.11}$$

This is of particular importance because the cross-correlation between various residues tells whether certain regions of the protein do or do not move in concert, which shows areas of differing flexibility in the protein. Similarly, the mean-square (ms) fluctuations of residue *i* can be determined by

$$\langle \Delta \boldsymbol{R}_i \Delta \boldsymbol{R}_i \rangle = \langle \Delta R_i^2 \rangle = \frac{k_{\mathrm{B}}\mathrm{T}}{\gamma} \mathrm{tr}(\boldsymbol{H}_{ii}^{-1}). \tag{2.12}$$

Calculating the ms fluctuations is important for several reasons. The first of which is because unlike with the cross-correlation value, the ms fluctuations yield a quantitative value of how much a particular residue is moving in space. Second of all, the ms fluctuations are easily comparable to experimental and MD B-factors because, minus a small scaling factor, they compare the same quantitative information (see B-factor discussion below for more information) [44].

The B-factors are compared to the ms fluctuations via their Pearson correlation coefficient (PCC), which is a measure of the strength of linear dependence between the two variables [45]. Doing so provides an unbiased comparison between the two data sets (experimental/MD to ENM results) and provides a definitive value of the fit between the two sets of results. This allows for a quick and easy assessment of how well each set of r_c and k recreates the fluctuations seen in the real physical system.

2.2.2 Debye-Waller (B) Factor

B-factors are used in describing the fluctuations of atoms from thermal vibrations [46, 47]. Derived from the attenuation of x-ray scattering, where B is the b-factor, q is the scattering vector and u(t) is the displacement of the scattering vector (as a function of time)

$$B = e^{-\langle [qu(0)]^2 \rangle} \tag{2.13}$$

As atomic coordinates are derived from x-ray crystallography, b-factors are also calculated from the x-ray scattering plots.

While the output from NMA does not directly give B-factors, the given output of ms fluctuations can be converted to B-factors by the relationship

$$B_i = \frac{8\pi^2 k_B T}{3\gamma} tr(H_{ii}^{-1}) \tag{2.14}$$

2.2.3 FEM Theory

The theory presented in the previous section allows for the determination of dynamic fluctuating properties of actin (i.e. mode shapes, resonant frequencies, etc…), however, information regarding the structural response of the system is also desired. Therefore, investigating the system via an FEM model is the best way to achieve this goal. Because the field of FEM is quite mature, this section will only give a cursory introduction to the theory beginning with an example of a simple, 1D spring (Figure 9). This example is still pertinent though because while FEM modeling can become quite complex with regards to choosing mesh type, size, etc…, this model only includes nodes connected by axially loaded linear spring members.

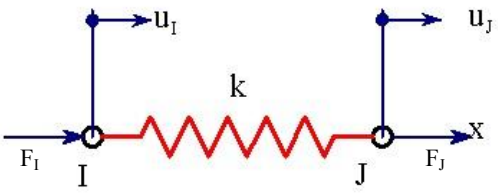


Figure 9. Simple 1D spring system with 2 nodes and forces applied at each node. Displacements at node I and J are labeled as u_I and u_J, respectively [48].

Continuing with the theory developed in the aforementioned section, the simple potential energy relationship from equation 2.1 allows for the development of the well known FEM element stiffness matrix. In order to arrive at Hooke's linearly elastic spring law, equation 2.1 needs to be integrated to yield the well known relationship

$$F = k(u_I - u_J) \tag{2.15}$$

From here, the element stiffness matrix for a single spring can be determined by decomposing the problem into a frame that tracks the displacement of each node [48], where

$$\begin{bmatrix} k & -k \\ -k & k \end{bmatrix} \begin{bmatrix} u_I \\ u_J \end{bmatrix} = \begin{bmatrix} F_I \\ F_J \end{bmatrix} \tag{2.16}$$

Now, combining the results of equation 2.16 with the Kirchoff interaction matrix (equations 2.7 and 2.8), we arrive at the overall system relation, in matrix form

$$\boldsymbol{KU} = \boldsymbol{F} \tag{2.17}$$

Where **K** is the overall stiffness matrix, **U** is the displacement vector that tracks every node, and **F** is the force vector that tracks forces applied to each node. Displacement boundary conditions are accounted for in **U**, and initial conditions involving applied loads are accounted for in **F**, the force vector [49].

Using a combination of the ENM in an FEM framework will allow for the extraction of vital structural information from the system, such as nodal displacements, and elemental stresses and strains. Upon this implementation, the aforementioned ENM now becomes the SENM.

2.2.4 CalculiX

The SENM calculations will be performed using an open source FEM program called *CalculiX*, developed by Guido Dhondt et al. primarily for thermo-mechanical applications [50]. The package is an attractive program because the syntax is almost

identical to that of the well known FEM package ABAQUS, yet has much less overhead in terms of computational costs. Furthermore, since *CalculiX* is an open source program, the scientific community has easy access to use and further develop the research in this field. *CalculiX* supports 1D linear axial springs, and uses an embedded program called *spooles* to solve the matrix problem. In addition, *CalculiX* also comes with a separate visualization tool called *GraphiX* that helps in visualizing the resultant variables, such as displacement or stress.

2.2.5 MD/SMD

The MD/SMD simulation results that are presented in this work have been generously contributed from Steven Kreuzer, a colleague at The University of Texas at Austin. The simulations have been able to take advantage of the state-of-the-art computational resources available from the Texas Advanced Computing Center (TACC). The open source MD simulation package GROMACS (GROningen MAchine for Chemical Simulations) is used in performing all of the MD simulations [51].

GROMACS uses the GROMOS93a1 force field when calculating particle interactions, and explicitly solvates the protein in water. While most of the 9mer actin system was adaptable to run in GROMACS, it did not have the necessary force field parameters embedded in the code to run a simulation with the ADP nucleotide. Therefore, Steven Kreuzer added a custom entry specifying ADP's characteristics to the code's parameter table so that the 9mer F-actin structure could be successfully simulated with the ADP nucleotide bound.

Each performed simulation (including the initial minimization, axial and torsional load cases) is run for a total simulation time of 10 ns. Structure positions and energy information are recorded every 1 ps, and the time interval between every dynamics calculation step is 2 fs. The structures used in the comparison between the SMD and SENM simulations are generated by taking the average positions of the atom coordinates over the final 2.5 ns of simulation time. Doing so increases the chances that the protein has reached a relatively stable structure before recording final position data.

2.3 EXPERIMENTAL DATA

Model validation is a critical step in figuring out the 'goodness of fit' of any proposed model. The model validation test for the 9mer actin SENM will include both a comparison to SMD results for the localized atomic details, and comparisons to constitutive properties that have been measured in *in vitro* experiments. Just as the scientific community continues to adopt the results of MD simulations in predicting biological processes, the same techniques will be used in predicting the deformation of F-actin under load. In addition, the recent advancements in experimental techniques within the last decade have allowed for more precise and accurate measurements for both overall molecular properties and properties at residue specific sites. Therefore, the use of MD simulations and experimental results deemed necessary and justified to make the most accurate assessment of the SENM's viability as a coarse graining model.

2.3.1 Actin Constitutive Properties

Analogous to the types of loading scenarios that this work subjects a 9mer F-actin segment to, researchers have been able to experiment with single actin filaments to

uncover some of its basic constitutive properties. Taking advantage of the recent advancements in experimental techniques, they have been able to accurately and repeatedly measure the axial and torsional stiffness of actin filaments. This section briefly discusses those findings.

2.3.1.1 Axial Stiffness

Two independent researchers, Liu et al. and Kojima et al., have recently been able to quantify the stiffness of actin under axial loading [17, 20]. Liu et al., using a novel technique involving micro-fabricated cantilevers, axially loaded actin filaments of various length from 0 to 230 pN (the maximal physiological load seen *in vivo*) [17]. They noticed that under a low loading regime (up to 50 pN), actin displayed a non-linear strain. At higher tensions though, actin follows the trend of a linearly elastic polymer (Figure 10). From the shown length versus tension plot, they were able to extract an axial stiffness of 34.5 ± 3.5 pN-μm/nm.

Kojima et al., using an experimental technique called nano-manipulation with microneedles, determined the stiffness of actin, both with and without the actin binding protein (ABP) tropoymosin bound to the filament [20]. For a single, tropomyosin free actin filament, Kojima obtained an axial stiffness of 43.7 ± 4.6 pN-μm/nm, which is in very close agreement to other published results.

2.3.1.2 Torsional Stiffness

Tsuda et al. has been able to directly measure the torsional rigidity of an actin filament using optical tweezers, achieving a torsional rigidity of 8.0 ± 1.2 X 10^{-26} Nm^2

[19]. This value has been confirmed true by back-calculating the axial stiffness of actin and matching those results to direct experiments.

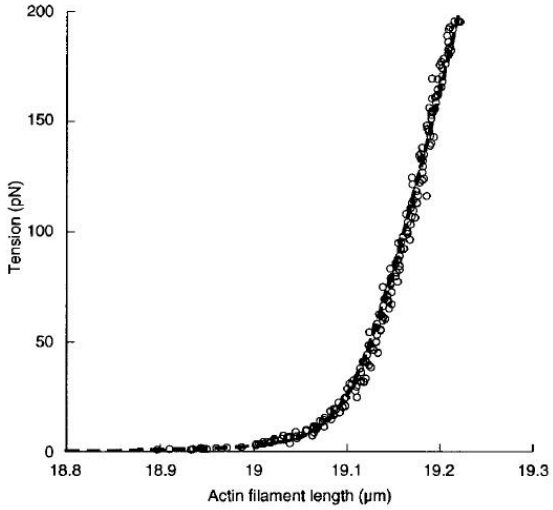


Figure 10. Actin length versus tension curve for a 19 μm long filament. A one stretch release cycle was performed over a period of 12 s using micro-fabricated cantilevers. Notice the F-actin non-linear loading response under 50 pN and linear response up to maximal physiological loading (230 pN). Actin filaments were polymerized *in vitro* from a 0.8 mM solution of G-actin [17].

That same year, Yasuda et al. also measured the torsional rigidity of actin filaments, GI_p, (where G is the shear modulus and I_p is the polar moment of inertia) with Ca^{2+} and Mg^{2+} bound to the high affinity binding site of actin. For F-actin-Ca^{2+}, a measured GI_p of 8.5± 1.3 X 10^{-26} Nm^2 was obtained, which corresponds very closely to the value from Tsuda [52].

2.3.2 Residue specific pulling experiments

To date, the work performed by Eyal and Bahar comes closest to that of the work proposed in this thesis. Eyal and Bahar, in an effort to study the anisotropic response of proteins to load, applied their developed ANM to experiments that pulled at residue

specific portions of a protein. Surveying proteins from ubiquitin to fibronectin, they were able to obtain the same relative sizes of forces needed to displace residues as those seen in experiments [25]. In fact, two proteins showed an experiment to ANM correlation of around 0.94. These results not only show the need to further the research in this field because of the preliminary work performed thus far, but they also help to validate the approach as a sound technique to extract localized (residue level) protein response to external loads.

2.4 EXISTING ACTIN MODELS

There have been many MD simulations run and course graining (CG) models, to include network models, developed for both G-actin and F-actin in an effort to better understand the conformation and dynamics of actin. The following section discusses these research findings and their conclusions.

2.4.1 MD simulations and Course Graining

Chu et al. performed MD simulations on G-actin, G-actin trimers, and 13mer sections of F-actin investigating the effect of the bound nucleotide on constitutive structural properties such as the persistence length, L_P (the length where a polymer transitions from behaving like a flexible elastic rod to more random fluctuating motions) [22]. With ATP as the bound nucleotide, F-actin behaved stiffer, exhibiting an L_P of 16 μm, where with ADP, the L_P is cut in half to around 8.5 μm. This confirmed the hypothesis that the shorter DB-loop found with bound ADP weakens the inter-monomer connections and causes a more flexible, disordered filament.

Zheng et al. performed multiple MD simulations on G-actin with various bound nucleotides and discovered that not only is the state of the DB-loop structure reversible, but there are additional loops in the G-actin structure that are affected by the nucleotide state [23]. These results show that the nucleotide state of G-actin is very important in determining the structure of the monomer, and that small changes (such as the dissociation of p_i, the inorganic phosphate produced after ATP hydrolysis), to the NBP

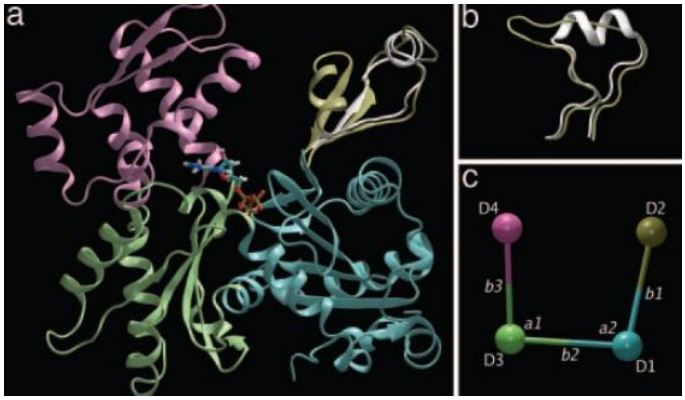


Figure 11. CG procedure employed by Chu and Voth where each SD corresponds to a ball of mass m (figure a/c). Properties such as SD-SD distance and dihedral angle were also input parameters into the model [22].

are amplified throughout the structure. In 2010, another MD simulation of F-actin was performed with a more modern filament structure. This study reconfirmed the conclusion that a large part of the F-actin flexibility is determined only by the structure of the DB loop (

Figure 11, b) [21].

Chu et al. also course grained G-actin to a level of that seen in

Figure 11, c. Sub-domain properties were gathered from MD simulations in [22], and filament vibration modes and constitutive properties such as force-extension curves

were successfully recreated with the model [53]. Their technique shows that to obtain constitutive polymer properties of F-actin, such as L_P and axial stiffness, a G-actin model course grained to include only 4 nodes is required. Similar dynamic results (vibrational modes) were achieved on a 13mer F-actin strand with a technique called the substructure synthesis method (SSM) [54]. A variation to their CG procedure includes generating the potential of mean force (PMF) from MD simulation results [55]. Compared to the harmonic potential used in the ENM, the PMF accounts for large scale conformational changes that commonly occur from the hydrolysis of a nucleotide as seen in G-actin.

While not applied directly to the F-actin system, Sept et al. has developed a CG technique applied to the cytoskeletal proteins α and β-tubulin found in microtubules [56]. Here, the MD simulation results from an unloaded microtubule structure were used to develop the CG model. The CG model was subjected to compression, bending and shear loads and accurately predicted the structural response as observed in experiments.

2.4.2 FEM models

Dr. Bathe developed a protein FEM model using the same mesh generation technique used in solid modeling where you take a solid body and discretize it as if it were an isotropic volume, as is commonly done on more conventional materials, such as metals and plastics [57]. The molecular volumes of the protein, in this case being T4 lysozyme, G-actin and a 52mer F-actin strand, were taken from their solvent extruded surface. From this model, the lower modes of the 3 aforementioned proteins were generated and consistently matched previously obtained results, including the F-actin strand, which was over 140 nm in length [57].

2.4.3 Additional Network Models

Since Tirion's early network model, which comprised of connecting atoms with Hookean like springs, there have been many iterations to further improve these network model's fluctuation correlations to experimental b-factors [42, 58]. As can be seen on the next page in Table 1, there exist many ENMs, each with their own set of advantages and disadvantages.

The first two, the GNM and ANM, contain a similar network in the sense that nodes are approximated from residue C_α's and all residues within a predetermined cutoff radius, r_c, are connected with springs that have the same spring constant. These networks are by far the simplest to implement because all the residues and bonds in the protein system are homogenized from the same two components.

The next three models (BENM, DNM and CNM) are all similar to each other because they differentiate between the types of bonds seen in real physical systems. Obviously, the covalent bonds that connect the backbone of proteins are going to be much stronger and stiffer than the hydrogen bonds and electrostatic interactions that

Table 1. ENM Comparison between various models

Model	Basic Assumptions	Advantages	Limitations/Difficulty	Online Server
Gaussian Network Model (GNM)	C_α level of resolution with uniform mass. Motion of residue is assumed to be isotropic. 1 model input: the cutoff radius (R_c). Spring constant, k, is scaled from B-factor fitting [40]	Simple to implement. Correlation b/w theory and experiment is 0.6-0.65*	Cannot decipher direction of residue motions (N X N matrix size)*	Yes, can be found at: http://ignm.ccbb.pitt.edu/GNM_OnlineCalculation.htm
Anisotropic Network Model (ANM)	C_α level of resolution, with uniform mass. 1 model input, R_c, and spring const., k, is scaled from B-factor fitting [39]	Simple to implement. Correlation b/w theory and exp. increases to 0.6-0.7. CAN determine direction of residue motion*	Anisotopy of residues requires adoption of larger R_c's, so more computational time is required (3N X 3N matrix size) – takes ~27 times longer to solve than N X N matrix*	Yes,at: http://ignmtest.ccbb.pitt.edu/cgi-bin/anm/anm1.cgi
Backbone Enhanced Network Model (BENM)	C_α level of resolution with uniform mass. Similar to ANM theory. R_c is 1 model cutoff, but k between backbone residues is stronger (by factor of 42). Optimum R_c similar to ANM [59]	Again, simple to implement. Correlation response is maximized when the backbone enhancing factor is increased by 42 (so $\gamma_{BB} = 42\gamma_{other}$). Correlation not explicitly state. CAN determine direction of residue motion*	3N X 3N matrix size (longer solving time)*	None found

Table 1. Continued

Model	Basic Assumptions	Advantages	Limitations/Difficulty	Online Server
Distance Network Model (DNM)	C_α level of resolution, uniform mass. R_c is set to predetermined values (2.3, 3.3, 5, 7, 9, 11 Å), with k scaled to each R_c depending on number of interactions for a given R_c. [60]	Correlation ~0.65. Possible to determine direction of residue motion*	3N X 3N matrix size (longer solving time)*	None found
Chemical Network Model (CNM)	C_α level of resolution, uniform mass. 2 types of residue connections, based on chemistry: 1) Backbone (BB), 2) Hydrogen bonding (HB). $k_{BB} = 10k_{HB}$. Rc is b/w 4.0 and 4.5 Å [61]	High correlations, ~0.70-0.75, but experimental B-factors were taken from a smaller sample of high-resolution PDB files (<1.0 Å resolution)*	Cannot decipher direction of residue motions (N X N matrix size). Isotropic motion is assumed.*	None found

* Refers to inferences made by author

occur between side chain interactions. These models attempt to account for these interactions by using either two spring constants (BENM and CNM) or many spring constants depending on the distance two residues are from each other (DNM).

The differences between these two sets of models serve as both advantages and disadvantages. The more realistic modeling seen by the nature of the residue interactions in the BENM, CNM and DNM models give higher normal mode correlation coefficients to experimental results than do the GNM and ANM. At the same time though, this adds complexity to the model and is more computationally expensive.

Therefore, choosing the ANM as the model system in this work is a combination of both the simplicity of the model along with the combination of the fact that the results of using any of the previously mentioned models in this type of application is currently unknown (aside from the work in [25]). While some of the aforementioned models seem to perform better in determining the dynamic response of unloaded proteins, it is still unknown how they would do under the proposed loading scenarios.

2.5 SUMMARY

This chapter has introduced the actin protein G-actin, and its filament structure F-actin. In addition, all pertinent background theory was presented that will be used in the calculations and results section that are included in Chapter 4. The current state-of-the-art in terms of protein structural analysis, MD simulations and coarse graining is presented, along with actin experimental data.

The next chapter covers the specific structure of F-actin to be used in the SENM simulations, and also discusses the methodology this work will apply towards the end goal of determining how load is transmitted through a structural protein. The features regarding each loading scenario applied to F-actin will be discussed in great detail.

Chapter 3

Model System and Methodology

3.1 Model System

As briefly introduced earlier in chapter 2, and as seen in Figure 12, the protein system to be investigated with the SENM is a 9mer strand of F-actin bound to the ADP nucleotide. While most other F-actin studies involve a 13mer protein ensemble, the 9mer length has been chosen for a number of reasons.

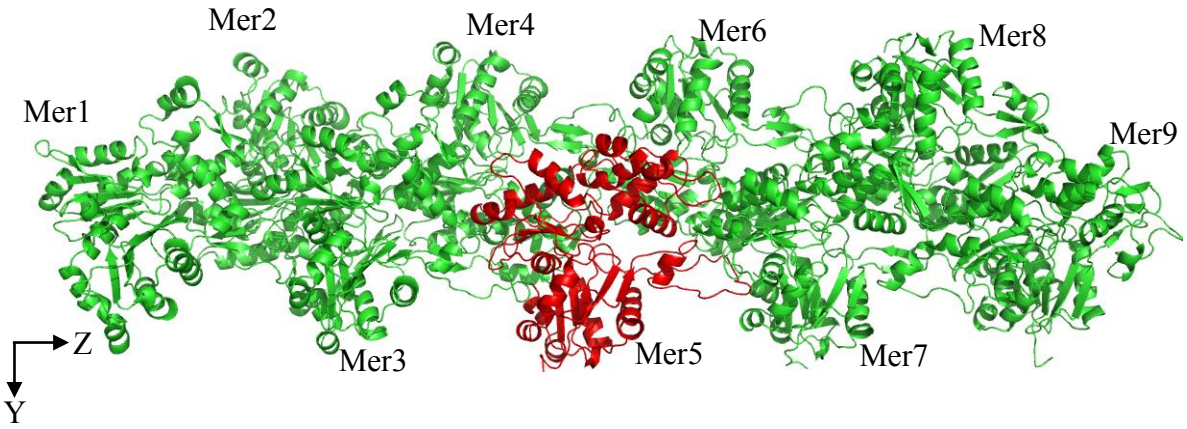


Figure 12. Ribbon representation of the 9mer F-actin structure to be used in the SENM model. The middle monomer, mer5, is highlighted in red (figure generated in PyMOL).

For the reason that the main purpose of this work is to better understand how F-actin deforms under various loading conditions that are seen *in vivo*, the F-actin structure in both the SMD and SENM simulations will have to be precisely and equivalently loaded. One reason is that inappropriate loading of F-actin could cause undesired and non-uniform stress/strain artifacts to propagate into mer5, the monomer under investigation. Therefore, using the theory behind St. Venant's principle, which states that

localized stress and strain distributions applied to the end of a body can be ignored at distances that are far from the end, allows us to approximate the loading between the SMD and SENM systems [62]. The use of this principle is also strengthened by the fact that the end loads will only be applied to monomers 1, 2, 8 and 9. Doing so will ensure that all of the monomers that directly bind to and interact with mer5 (mers 3, 4, 6 and 7) will not have any external loads applied to them.

Another important reason in choosing a 9mer system over the 13mer system is because of computational costs. The all atom 9mer F-actin system has 33,408 atoms, which is quite a large computational burden as the system lies; when considering the fact that the protein usually only accounts for 5-10% of the simulation size in a fully, explicitly solvated system, it can easily be seen how the size of the system grows quite rapidly. Increasing the system to 13 monomers increases the protein size alone by approximately 50%, which stretches the current computational resources too thin.

3.1.1 9mer F-actin Structure

The structure to be used in the SMD and SENM simulations stems from the work of Oda et al., who recently published an updated structure of G-actin while in the filamentous state (PDB code 2zwh) [32]. Using the structure generated from colleague Jun Zhou, who applied a transformation procedure to map the G-actin monomers into the F-actin conformation, the newly created PDB file (Figure 12) was then fed into GROMACS so that an energetically minimized, equilibrium structure could be determined. This newly minimized structure served as the starting structure for both the SMD and SENM simulations.

It is important to mention here that all of the MD simulations are performed with F-actin bound to the ADP nucleotide. Although a single F-actin can be found with all 3 nucleotide states present at any given time (ATP, ADP-P_i and ADP), ADP was used because an F-actin is predominantly found to have ADP bound in the NBP [63].

3.2 METHODOLOGY

The methodology presented discusses the steps this work takes to decipher the effects of load on the F-actin structure. Due to the unknown effects of load on a structure, additional steps are taken in the model parameterization before the structure can be axially and torsionally loaded. As seen in Figure 13, the process to determine the optimum parameters for the SENM model follows a circular flow. Beginning with a minimized, energetically equilibrated 9mer structure from the MD simulations, the SENM model is generated via the steps that follow the clockwise flow, including the preprocessing steps of removing the C_α's from the structure and generating the ENM from a given set of r_c and k. The counter clockwise flow shows the steps involved in the SMD simulations, which essentially comprises of applying the loading conditions to the structure, depending on the applied loading scenario. The two flows meet for the residual calculation (middle item in the bottom row), from where the next r_c and k parameters are chosen for the subsequent SENM iteration. Once the entire r_c versus k field has been scanned, the optimum parameters are chosen so that the model can be used for prediction.

3.2.1 Unloaded 9mer ENM Parameterization

One of the key initial questions this thesis is trying to answer is if one can use the optimum parameters derived from the unloaded NMA (r_c and k) and feed those back into

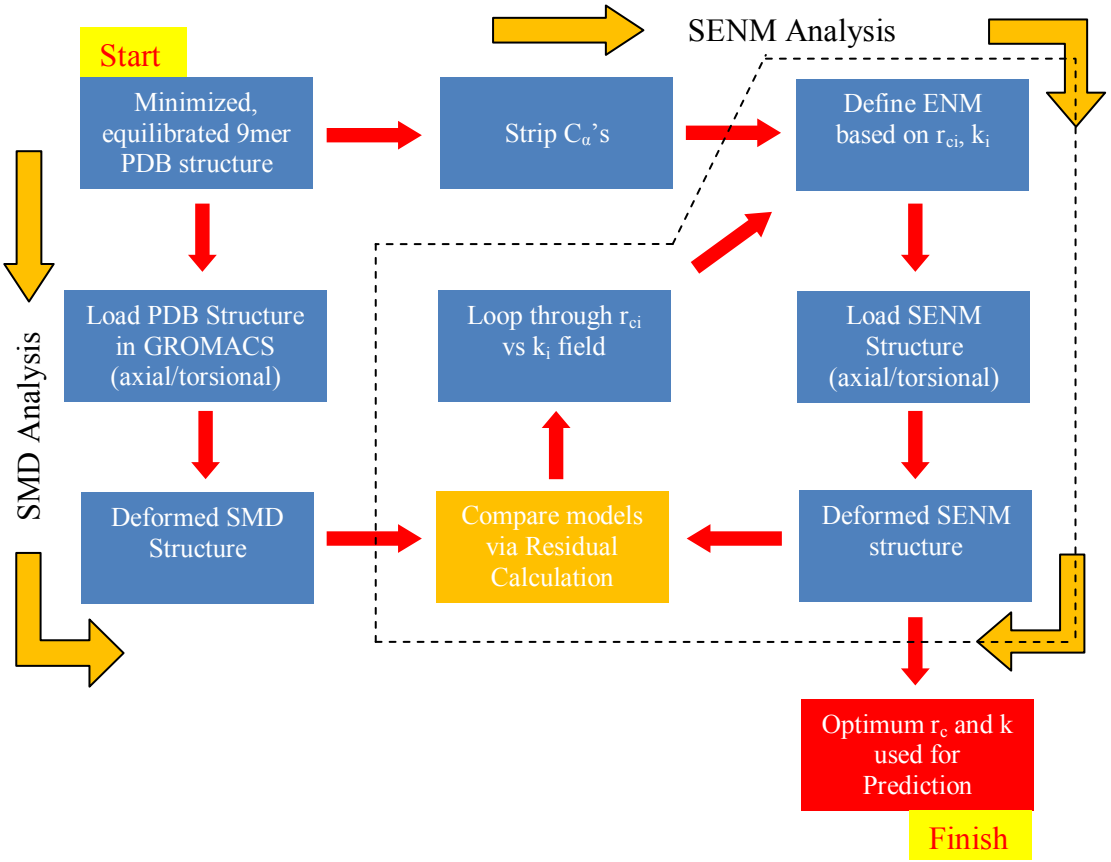


Figure 13. Flowchart showing the methodology in determining the optimum r_c and *k* parameters so that model predictions can be made. The chart starts with the energetically minimized, equilibrated 9mer structure from the MD simulations, and loops repeatedly until the r_c vs *k* field has been scanned. The processes enclosed in the dashed area has been automated with code generated by Dr. Liu (Appendix D).

the structural actin model under various loading scenarios with the goal of achieving the best model correlation to experimental and SMD results. While the answer is not intuitively clear, structural mechanics tells us that when a system is under load, whether it be axial, torsional, etc… the system will stiffen up. Here, a stiffer system implies a different frequency response behavior, including altered mode shapes and increased resonant frequencies [64].

However, it is not currently known if a deformed system parameterized with r_c and *k* values from the unloaded system will still optimally match the SMD results. Therefore, the first goal of this study is to determine the optimum r_c and *k* values for the unloaded 9mer system. In addition, because B-factors from crystallographic experiments have proved unreliable for ENM model parameterization [65], the B-factors from the 9mer MD simulations are used instead for the comparison to the unloaded ms fluctuations.

Using the source code from the online server provided by the University of Pittsburgh's Department of Computational Biology [66], colleague Esfandiar Khatiblou modified the code in a way that allows the program to compare the ENM ms fluctuations to MD B-factors, as opposed to its current method of using the experimental B-factors included in actin's PDB file). The modified code is attached in Appendix A. As explained in the theory of section 2.2.1, the code aims to match the model fluctuations to the MD derived B-factors and quantify the fluctuation match by means of the Pearson correlation coefficient (PCC).

3.2.2 Decoupling r_c and k

The theory presented in section 2.2.1 shows that for the unloaded, dynamic vibrational analysis of a protein structure, the model depends on only 1 parameter, r_c. Once r_c is determined, *k* is merely derived from the scaling of the ms fluctuations to match the b-factors. However, it is not known if the relationship between r_c and *k* for a loaded structure is the same as its unloaded counterpart. It is possible that due to the varying structural stiffness associated with an external load, the optimal r_c and *k* for one

loaded structure will match to that of another loaded structure. Therefore, when determining which combination of the r_c and *k* parameters are optimal for an actin filament under load, r_c and *k* will be decoupled so that the entire r_c versus *k* plane space can be scanned using a brute force method (Figure 14). The metric that will determine how well r_c and *k* match the SMD b-factors, termed the *residual*, is discussed later in section 3.2.5.

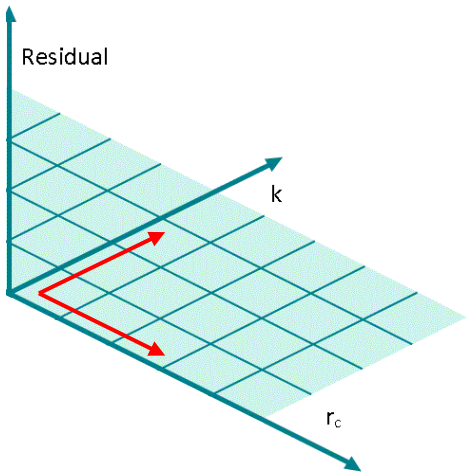


Figure 14. Image of the r_c versus k plane space which will be scanned to generate the optimum parameters for the SENM model based on the residual. Image edited from [67].

The initial upper and lower bound estimates for the r_c and *k* plane will begin with values derived in the unloaded F-actin parameterization. While the values of *k* can vary widely due to the hyperbolic behavior of the unloaded r_c versus *k* relationship, r_c has yielded good PCCs anywhere from 8.0 to 16.0 Å. Accordingly, once the r_c versus *k* results begin to come in and the resulting *residual* surface develops, the r_c and *k* ranges will be adjusted as necessary to account for any insufficient ranges that may have been initially used.

Colleague Dr. Liu has generously developed an extensive FORTRAN 90 script that automates the process scanning the r_c versus *k* field, along with integrating the FEM code and the protein fitting algorithm (discussed in Model Validation, Chapter 3.2.5 below). Now, the size of the scanned r_c versus *k* field can be easily modified with little additional input from the user. The FORTRAN90 code for this automation process is included in Appendix D.

3.2.3 Axial Loading

The first loading scenario actin is subjected to is that of an axial load. Of the two loading scenarios actin will have imposed on it, axial loading is by far the more important load actin will carry because it is the main method of load transmission observed *in vivo* [68]. Consequently, the results of this SENM simulation will yield a wealth of knowledge about actin's physical characteristics under load.

As mentioned before, St. Venant's principle will allow us to approximate the end loading for F-actin so that it does not have to be loaded exactly as it is *in vivo*, which would be extremely difficult to fully execute. However, in an effort to still load actin as realistically as possible, the loads will not be applied axially to actin, but instead will be applied along the vector connecting mers 2 and 8, and mers 1 and 9 (see Figure 15).

Each monomer has a 100 pN load evenly distributed to each atom (for SMD) or residue (for SENM) over the entire monomer, to yield a total axial load of 200 pN applied to each end, roughly the highest physiologically load seen *in vivo* [17]. In addition, so that no rigid body translation occurs to the protein system, GROMACS

automatically re-centers mer5 (and hence the entire structure) after each simulation iteration.

However, in the FEM package *CalculiX*, it is not possible to specify a region that will not have any rigid body motion without pinning the entire structure down as a node with zero displacement (and consequently add additional internal constraints and reactions to the system). To combat this issue, a pin joint was added at the midway point along the vector line connecting the center of mass (CoM) of mer2 and mer8, specifically at residues 1188, 1189 and 2164, which correspond to residues 63 and 64 of mer4 and residue 289 of mer6, respectively. While in order for the pinning location to be a true pin joint it should ideally have only one residue with restricted motion, doing so causes a local stress singularity in the FEM software, so the code would not run to completion. That is why the minimum of 3 adjacent residues that are not on mer5 (the monomer under

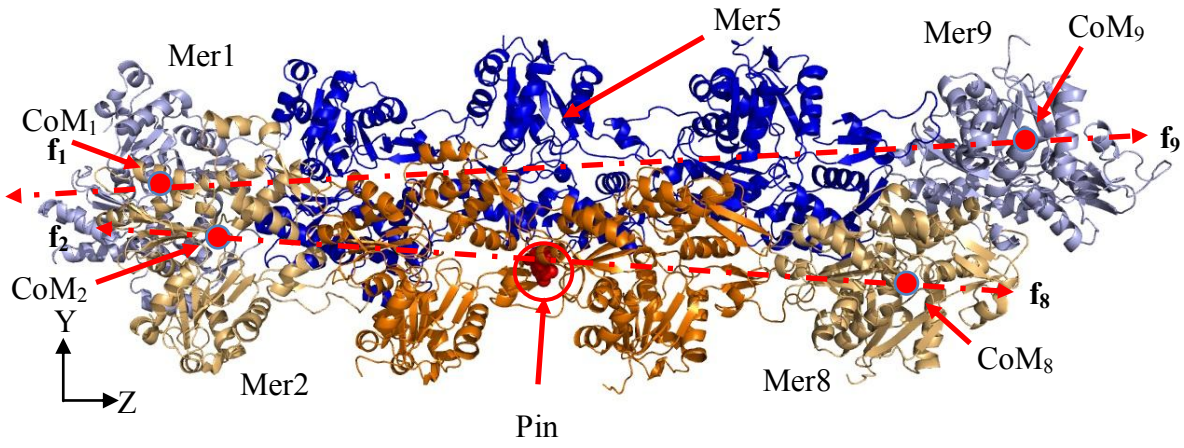


Figure 15. 9mer F-actin structure subjected to an axial load. The G-actins along the mer2-mer8 helix chain are colored orange and the mer1-mer9 helix chain actins are colored in blue. The loads ($\mathbf{f}_{1,2,8,9}$) are applied along the vectors that run from the CoM of mer1-mer9 and mer2-mer8, colored in red. The 3 pinned residues necessary in the SENM simulation are highlighted as red spheres along the mer2-mer8 vector (figure generated in PyMOL).

investigation), were chosen as then pinning location. The positions chosen to pin the structure also correspond to the point where the moments from the applied forces vanish. The *CalculiX* input file code is included in Appendix E.

3.2.4 Torsional Loading

The SENM 9mer structure will be torsionally loaded in a similar manner proposed by Steven Kreuzer and executed in the SMD simulations [69]. Smaller subgroups on the loading monomers (mers 1, 2, 8 and 9) are formed that are farther away from the filament axis; this helps to create a larger lever arm for torque application. The specific details of which residues are loaded on each monomer can be seen in Figure 16 and Table 2. As seen in the figure, the highlighted subgroups lie on the outer perimeter of the filament, and vary in size and region along the protein structure.

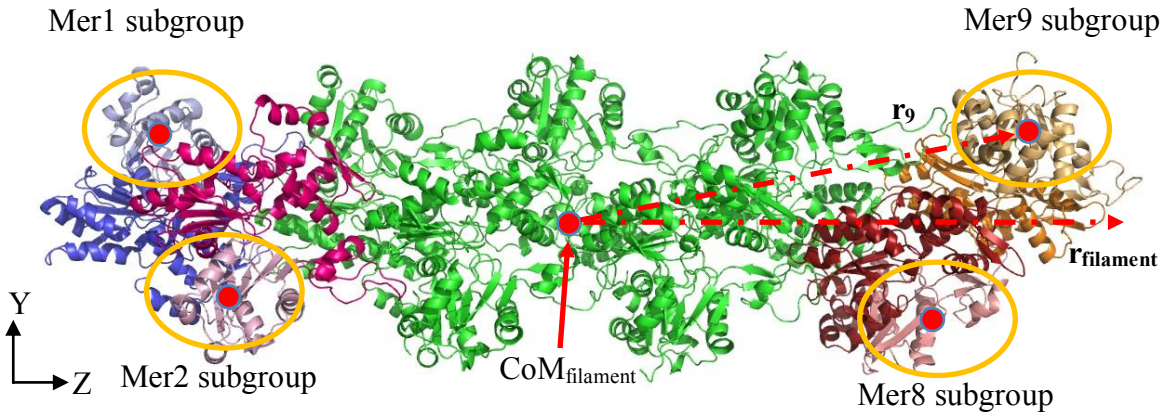


Figure 16. 9mer F-actin torsional loading locations. The filament CoM along with $\mathbf{r}_{\mathbf{filament}}$ are shown. $\mathbf{r}_{\mathbf{9}}$ is shown as an example $CoM_{filament}$ to subgroup vector. Taking the cross product of the $\mathbf{r}_{\mathbf{filament}}$ and $\mathbf{r}_{\mathbf{i}}$ vectors determines the loading direction on subgroup **i.** Figure adopted from [69] (figure generated in PyMOL).

Monomer	Global no.	Local no.
A	1-34	1-34
	68-136	68-136
	349-375	349-375
B	376-408	1-33
	446-511	71-136
	713-750	338-375
H	2626-2761	1-136
I	3001-3146	1-146
	3333-3375	333-375

Table 2. Global and local numbering of residues that define each monomer subgroup under torsion load [69].

In order to determine the direction of each applied torque, vectors from the $CoM_{filament}$ to the CoM of each loading subgroup, $\mathbf{r_1}$, $\mathbf{r_2}$, $\mathbf{r_8}$ and $\mathbf{r_9}$, were calculated, in addition to the vector that runs down the filament axis, $\mathbf{r_{filament}}$. Taking the cross product of $\mathbf{r_{filament}}$ with each loading subgroup vector ($\mathbf{r_1}$, $\mathbf{r_2}$, $\mathbf{r_8}$ and $\mathbf{r_9}$) yields a vector, $\mathbf{r_{load\text{-}i}}$ that is perpendicular to $\mathbf{r_{filament}}$ (see Figure 17) [69].

A total torque of 200 pN-nm is applied to the filament, or 100 pN-nm per monomer subgroup. This torque value is derived from the hypothetical situation where myosin II head attaches transversely to an actin filament and applies a maximal torque [70] . On the actin filament, a torque of this magnitude yields a twist of around 3.6° at the outer radius of the filament, which corresponds to a displacement of around 2.2 Å (when F-actin is approximated as a cylinder). This displacement is on the same order of magnitude as seen in the axial loading simulations, which is the main reason a torque of

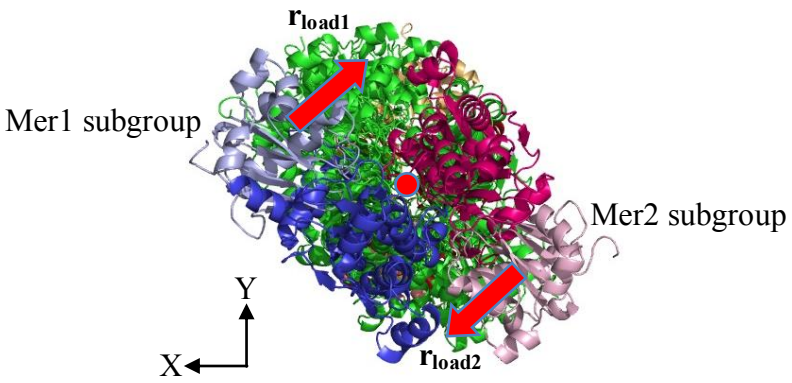


Figure 17. View looking down the filament axis. The torsional loading vectors, $\mathbf{r}_{load\text{-}I}$, can be seen for the mer1 and mer2 subgroups. The loading vectors for mer8 and mer9 subgroups are in the opposing directions (not shown) [69] (figure generated in PyMOL).

this size was chosen (see Appendix A for back-of-the-envelope axial load calculations and Appendix C for analogous torsional load calculations).As mentioned in section 3.2.2 that discusses axial loading, for the SENM simulations, the structure will be pinned at the same locations on mer4 and mer6 (residues 1188, 1189 and 2164). See Appendix F for the torsional loading *CalculiX* input file code.

3.2.5 Residual/Model Validation

Before detailed observations about the F-actin system under load can be discussed, the SENM needs to first be validated. It is important that the optimum model parameters (r_c and k) are chosen for each loading scenario so that F-actin observables are extracted from the most accurate model possible. In order to find the best model possible though, it is first necessary to match the SENM to the SMD simulations based on a predefined *residual*.

The residuals that will be used to compare the two structures are, for axial loading, a combination of the Dali (**D**istance Matrix **ALI**gnment) score and the change in length of the entire filament, termed $\Delta L_{19}/L_{19}$; for torsional loading, only the Dali score will be used. As augmented by colleague Dr. Liu, additional residual metrics are proposed based on the quantities of interest (QoIs), as discussed ahead in section 3.3, which include $B_1/\Delta B_1$, $B_2/\Delta B_2$, $D_{inter}/\Delta D_{inter}$ and $\varphi_d/\Delta\varphi_d$ These quantities of interest are calculated in a similar manner to the $\Delta L_{19}/L_{19}$ equation (equation 3.2).

The Dali score is a popular technique for pair wise structure comparison that uses four metrics to determine an overall cumulative value for how well two structures match spatially. These aforementioned metrics are the Z-score (which, from statistics, measures the deviation from the mean of a weighted sum of similarities of intra-molecular distances), the number of aligned residues between the two structures, the RMSD of C_α's, and the sequence identity between the two chains [71, 72]. In this case though, the last metric will not make any difference in the Dali score because the two compared structures have the exact same sequence. The RMSD of C_α's is already a widely used technique to compare backbone conformation of structures [23, 73], and the RMSD provides a simple, yet effective metric in quantifying the comparison between the SENM and SMD structures. As seen in equation 3.1, if vectors $\mathbf{v_i}$ and $\mathbf{w_i}$ point to the location of residue **i** on each structure, then the RMSD for the whole structure is calculated by:

$$RMSD = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\|v_i - w_i\|^2} \tag{3.1}$$

where the counting variables **i** through **N** represent the residues over which the RMSD value is calculated.

The other metric that will be used in calculating the fit between the SMD and SENM structures for axial loading is the $\Delta L_{19}/L_{19}$ distance [74]. While this metric does not have the residue-level resolution as presented with the Dali score, it gives an easy way to determine if the overall system deformation of the SENM structure is closely matching the SMD results. In the $\Delta L_{19}/L_{19}$ metric, L_{19} represents the SMD CoM distance between mer1 and mer9, and ΔL_{19} is the difference between the SMD and SENM deformations under axial load, where

$$\Delta L_{19} = \Delta L_{19SMD} - \Delta L_{19SENM} \tag{3.2}$$

The other aforementioned QoI metrics, including $B_1/\Delta B_1$, $B_2/\Delta B_2$, $D_{inter}/\Delta D_{inter}$ and $\varphi_d/\Delta\varphi_d$ [74], are calculated in a manner similar to equation 3.2. These residual metrics allow for a more in depth assessment of how well the SENM model is matching the results from the SMD simulations.

The procedure for generating the Dali score, $\Delta L_{19}/L_{19}$ distance, and additional QoI residual metrics requires a few steps. First of all, because the SMD simulations are very dynamic in nature, the structure can undergo small rigid body translations and rotations that will drastically affect the residuals if matched as is. Therefore, a protein least-squares fitting program using the McLachlan algorithm called ProFit (developed by Dr. Martin's Lab at the University College London) is used to match the two structures and removes any rigid body rotations and translations that may have occurred during simulations [75].

One option from ProFit that is taken advantage of is the ability to specify what areas of the protein need to be fit. Because we are specifically interested in the deformations that occur in the middle, or 5th monomer, ProFit will first fit these sections to each other as closely as possible, and then the remaining parts of the structure will be fit (as opposed to fitting the entire structure without any preference to a particular section).

3.3 QUANTITIES OF INTEREST

In addition to presenting and comparing the displacement of each residue from the SENM with the SMD results, four additional quantities of interest (QoIs) will be presented that better describe the localized effects mer5 sees from the axial and torsional loads. These four QoIs include: B1, B2, the dihedral, or torsion, angle φ_d, and the intermer distance between mer5-residue 288 and mer7- residue 204.

3.3.1 B1 and B2

The quantities B1 and B2 were developed by Dalhaimer et al. in an effort to better quantify the NBP state in a G-actin monomer based on the distances between key residues in that region [28]. As seen in Figure 18, the distance B1 is defined as the length between the C_α of residue 14 in SD1 (blue) to the C_α of residue 158 in SD3 (blue). Also shown in Figure 18 is B2, the distance defined by the length between residue 15 in SD1 (green) to residue 157 in SD3 (green). Officially, the length of B2 is used to define the state of the NBP, which is said to be open if B2 > 7.0 Å, closed if B2 < 6.0 Å, and in an intermediate state if 6.0 Å < B2 < 7.0 Å [28].

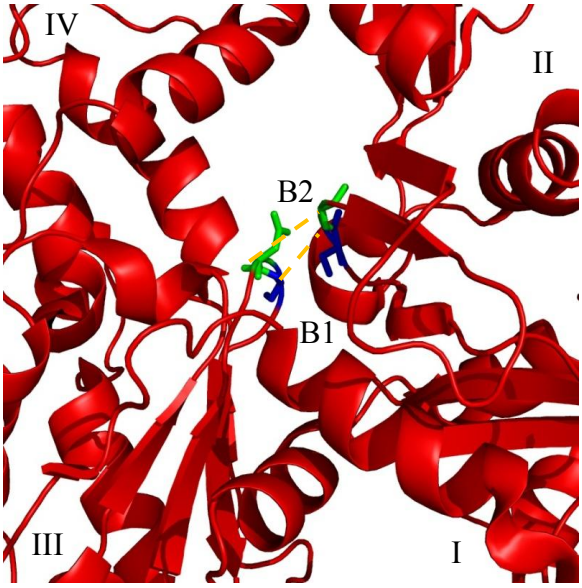


Figure 18. Close up of the NBP showing the distances B1 and B2 which characterize the state of the pocket. The distance B1 connects residue 14 of SD1 to residue 158 of SD3 while B2 connects residue 15 of SD1 to residue 157 of SD3 (figure generated in PyMOL).

3.3.2 The Dihedral Angle φ_d

While a residue level description of mer5 is lost in studying the dihedral angle, we can still learn a lot about the stress the monomer is under by this QoI. Figure 19 shows a schematic of a G-actin monomer, and each subdomain's coarsening into its CoM position. Here, φ_d is defined as the angle between the vector that points from the CoMs of SD1 to SD2 and from SD3 to SD4, and is calculated by

$$\varphi_d = \cos^{-1}(\overrightarrow{n_{134}}.\overrightarrow{n_{213}}) \tag{3.3}$$

where $\overrightarrow{n_{134}}$ is the vector normal to the plane made from SDs 1, 3 and 4, and $\overrightarrow{n_{213}}$ is the vector normal to the plane from SDs 2, 1 and 3.

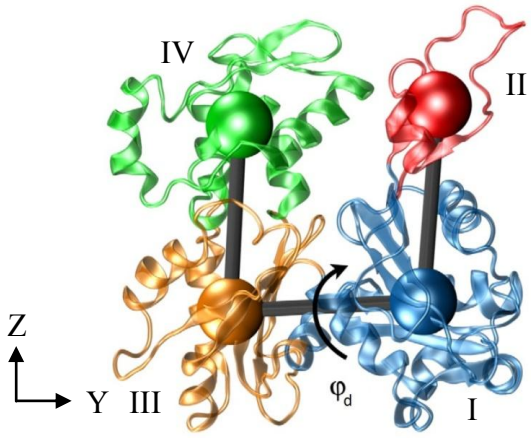


Figure 19. Image of a G-actin monomer, and its relation to the dihedral angle, φ_d. Each colored ball represents the CoM of a SD.

Calculating φ_d is important in characterizing the G-actin monomer for a number of reasons. First of all, it is a metric that is straight forward to calculate in both the SENM and SMD structures. Secondly, it gives us an idea of the amount of rotation between SD2 and SD4. As was presented by Oda et al., G-actin is observed to rotate by as much as 20° during the polymerization process [32]. Knowing this value gives insight as to the relative amount of rotation a monomer in the filament is subjected to under various loading conditions.

3.3.3 Intermer Distance

In another effort to quantify how well the SENM model matches the deformations observed in the SMD simulations, the intermer distance best achieves this task. This QoI will determine the displacement between mer5 and its adjacent neighbor mer7 at the residue level.

This metric compares the distance between the identical residue numbers of adjacent monomers, specifically residue 288. As seen in Figure 20, residue 288 was chosen for this QoI because it lies on the boundary of the monomer; therefore we can garner a better understanding of how mer5 stretches the line acting between these two residues.

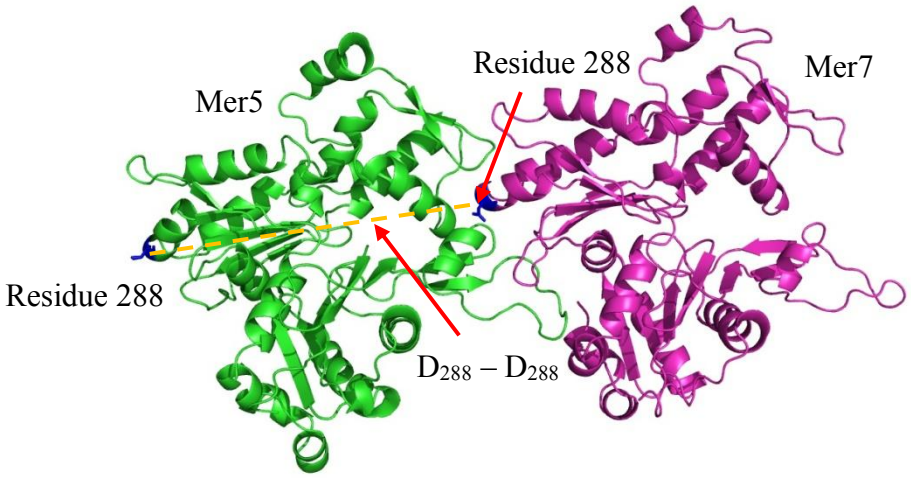


Figure 20. The QoI showing the distance between residue 288 (blue) of mer5 (green) and residue 288 (also blue) of mer7 (magenta), labeled as $D_{288} - D_{288}$ (figure generated in PyMOL).

3.4 SUMMARY

This chapter has discussed how, using the many available open source tools from the scientific community, this work has synthesized a methodology to uncover the underlying structural details that are occurring in a protein under load. Beginning with an energetically minimized, equilibrated 9mer F-actin structure, SMD simulations are performed in parallel with the SENM simulations under a range of r_c versus k values. The two simulations will subject the F-actin structure to equivalent axial and torsional loads. The resulting structures are post-processed with ProFit, and the parameters that

best fit the SMD simulations are used to analyze the effectiveness of the SENM model. The upcoming chapter discusses the results from these simulations.

Chapter 4

Model Parameterization and Loading Results

4.1 INTRODUCTION

This chapter discusses the results of the SENM model parameterization and results from the F-actin loading under various boundary conditions. For each loading scenario, the results from the model parameterization are discussed to show its sensitivity to r_c and *k*. Then, the residue displacements are presented for the middle mer, or mer5, of the structure. This chapter concludes with a table summarizing the QoIs for the two loading scenarios, and a discussion of the implications of the overall modeling results. It should be noted here that in the following sections discussing residue displacements, it is actually the *magnitude* of the residue displacement that are being calculated and referred to.

4.2 UNLOADED MODEL PARAMETERIZATION

Because the online server from the University of Pittsburgh cannot process structures as large as a 9mer actin filament, the unloaded model parameterization was performed on machines at TACC. Using these state-of-the-art resources still pushed the memory limits of these computers and proved very computationally expensive; therefore the resolution of the r_c step size is at the half-angstrom level. With an r_c range from 10 to 20.0 Å, we are still able to find the optimum r_c value that coincides with the point where the PCC is at a maximum.

The resulting plot seen in Figure 21 shows the spring constant, *k*, and PCC for the entire 9mer structure. There are two main messages that can be gathered by looking at these results; the first is that the optimum r_c, corresponding to the point of maximum PCC, occurs at around 12.0 Å (here, *k* is 17.9 pN/Å). The second is that *k* follows a hyperbolic trend, theoretically increasing to ∞ as r_c approaches 0 and decreases to 0 as r_c approaches ∞ (as determined from a least squares fit where $R^2 = 0.96$). This is due to the fact that as r_c increases, the number of elements in the system increases quadratically, so *k* needs to decrease appropriately to maintain the same overall system stiffness (trend not shown).

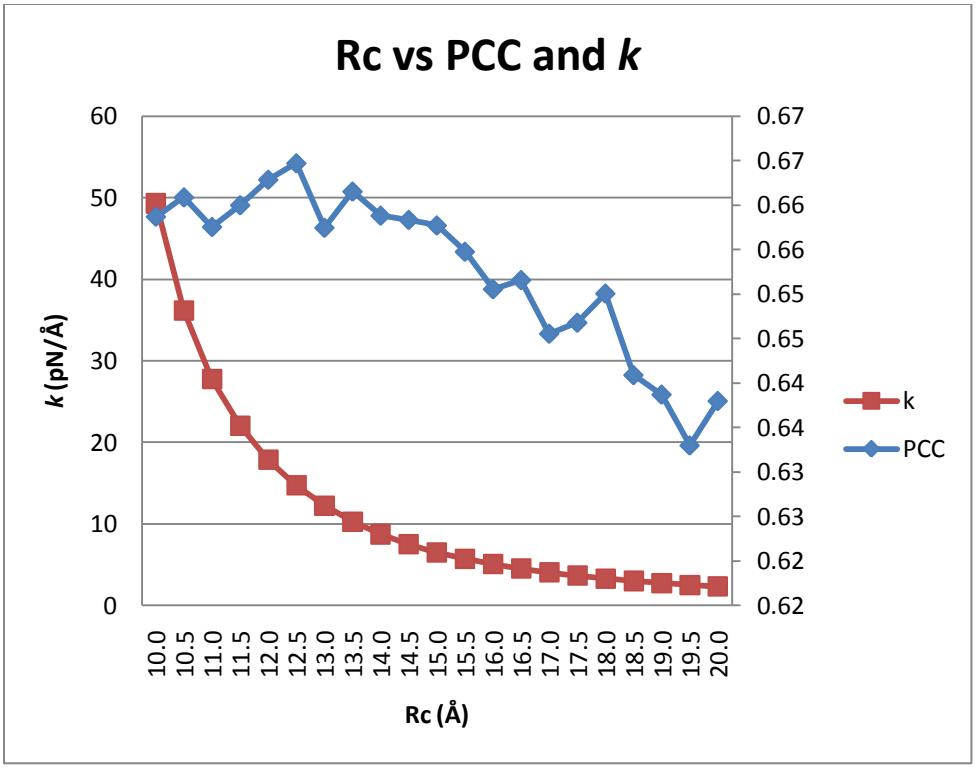


Figure 21. Unloaded r_c versus k and PCC for the 9mer structure under normal mode analysis [76]. Notice how the spring constant displays a hyperbolic trend and decreases to 0 as r_c approaches ∞. The PCC reaches an overall maximum value of 0.67 when r_c is 12.5 Å.

4.3 AXIAL LOADING RESULTS

Upon completion of the SMD simulations and FEM modeling of the 9mer structure in *CalculiX*, the axial parameterization and loading results are presented and discussed in the context of the overall, regional displacements of the middle monomer, mer5.

4.3.1 SMD Axial Results

After the 200 pN axial load was applied to the minimized MD structure, the simulation was run for a total time of 10 ns, and the displacement coordinates were taken from the average of the final 2.5 ns of simulation time. The data presented in this section is from simulations performed by Steven Kreuzer. [69].

As can be seen in Figure 22, certain regions of the G-actin monomer deform much higher than others, including the DB-loop and the loops around residues 220-235 and 240-248. In addition, the aforementioned loops, as seen in Figure 23, correspond to regions of G-actin that bind to neighboring residues. To put the magnitude of these displacements in context, it should be noted that the RMSD of mer5 alone is approx. 2.78 Å, which shows that there are a lot of fluctuations and variations in the G-actin structure to begin with.

Residues 322-325, which are also known to bind to its adjacent neighbor's residues 243-245, displaces itself by 7.0 Å, meaning that this region transmits a lot of load through this binding site. The N-terminus of the chain around residues 2-5 also

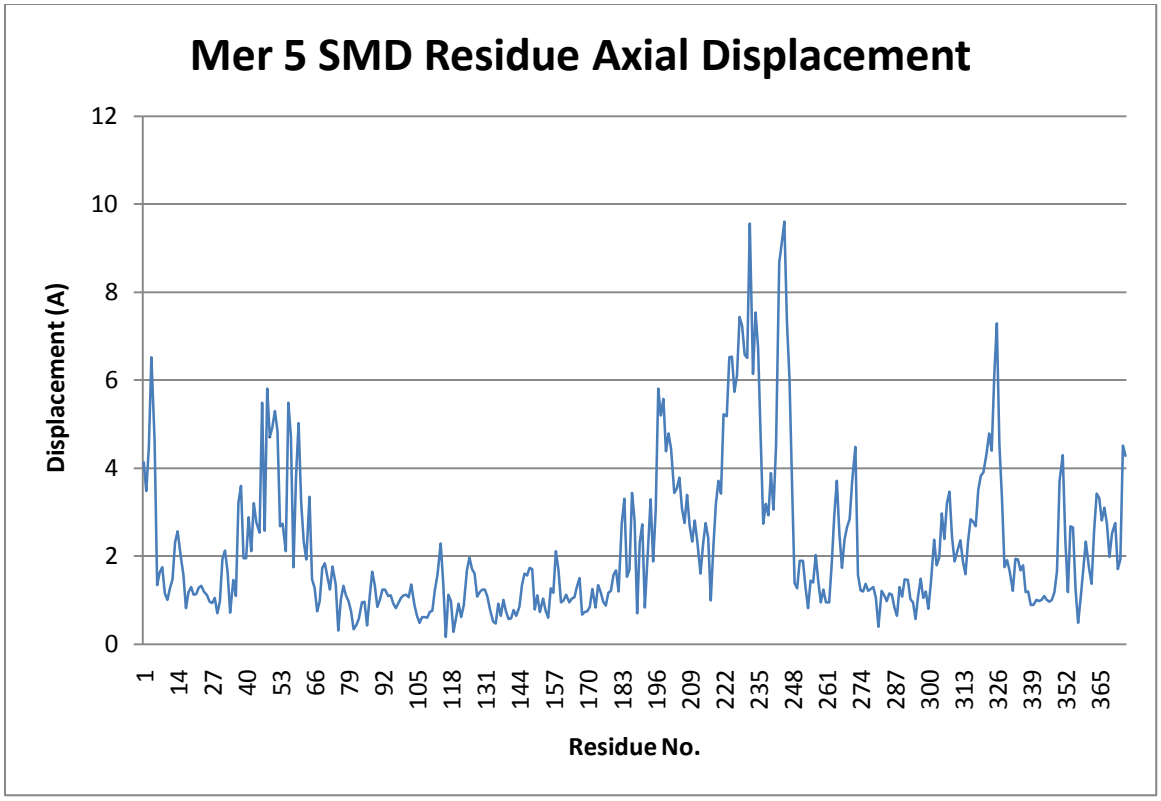


Figure 22. Mer5 residue displacements from a 200 pN axial load. The displacements are in Angstroms, with some residues moving distances up to 1 nm. Data from [69].

undergoes a rather large displacement, yet this region is not known to bind to any neighboring monomers. This is likely due to the fact that because this region of G-actin is unconstrained by any inter-residue interactions, this end is very susceptible to molecular vibrations caused by thermal fluctuations. Another unforeseen area with large displacements occurs around the loop by residues 223-230 (seen as the yellow and orange α-helix on the left hand side of Figure 23). It is likely that because of the geometry of that loop, and its protrusion from the existing structure (and hence its lack of binding to neighboring residues), it is also very sensitive to the thermal fluctuations that are occurring in the MD simulations.

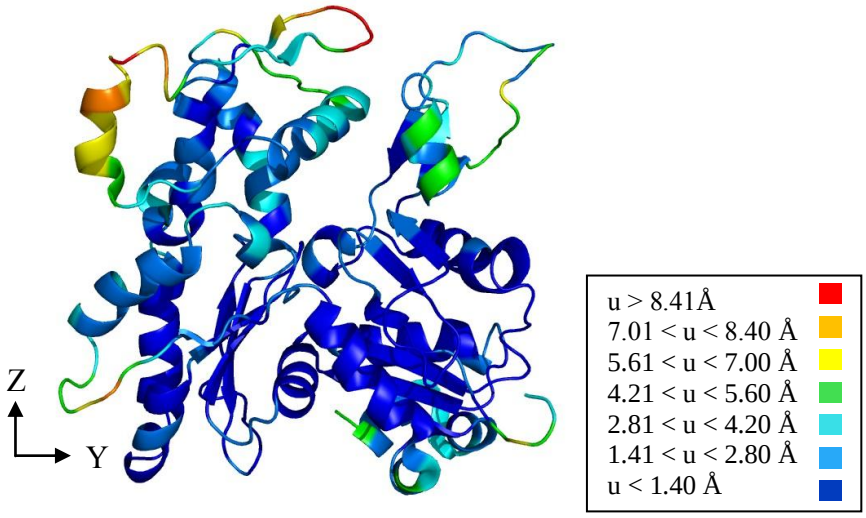


Figure 23. Ribbon representation of G-actin, colored according to the magnitude of SMD residue displacements. Regions of lower displacement are represented by cooler colors (blue, marine, cyan) while regions of higher displacement are displayed as warm colors (yellow, orange, red). Note how the loop around residues 240-248 undergoes displacements larger than 8.4 Å. Data from [69] (figure generated in PyMOL).

4.3.2 Axial Load Parameterization/Validation

Using the automation code provided from Dr. Liu [74], a total of 6 metrics were used in an attempt to parameterize the SENM model. For axial loading, there are two residual values that proved successful in finding the optimum r_c and k values for the SENM model. The first of one, the Dali score, is used extensively in the structural comparison of proteins; the second one, the $\Delta L_{19}/L_{19}$ metric, is useful because it is a straightforward calculation that matches the two structures based on their overall constitutive stiffnesses. The other 4 metrics that were derived from the OoIs proved unreliable in forming a consensus on the optimum r_c and k parameters.

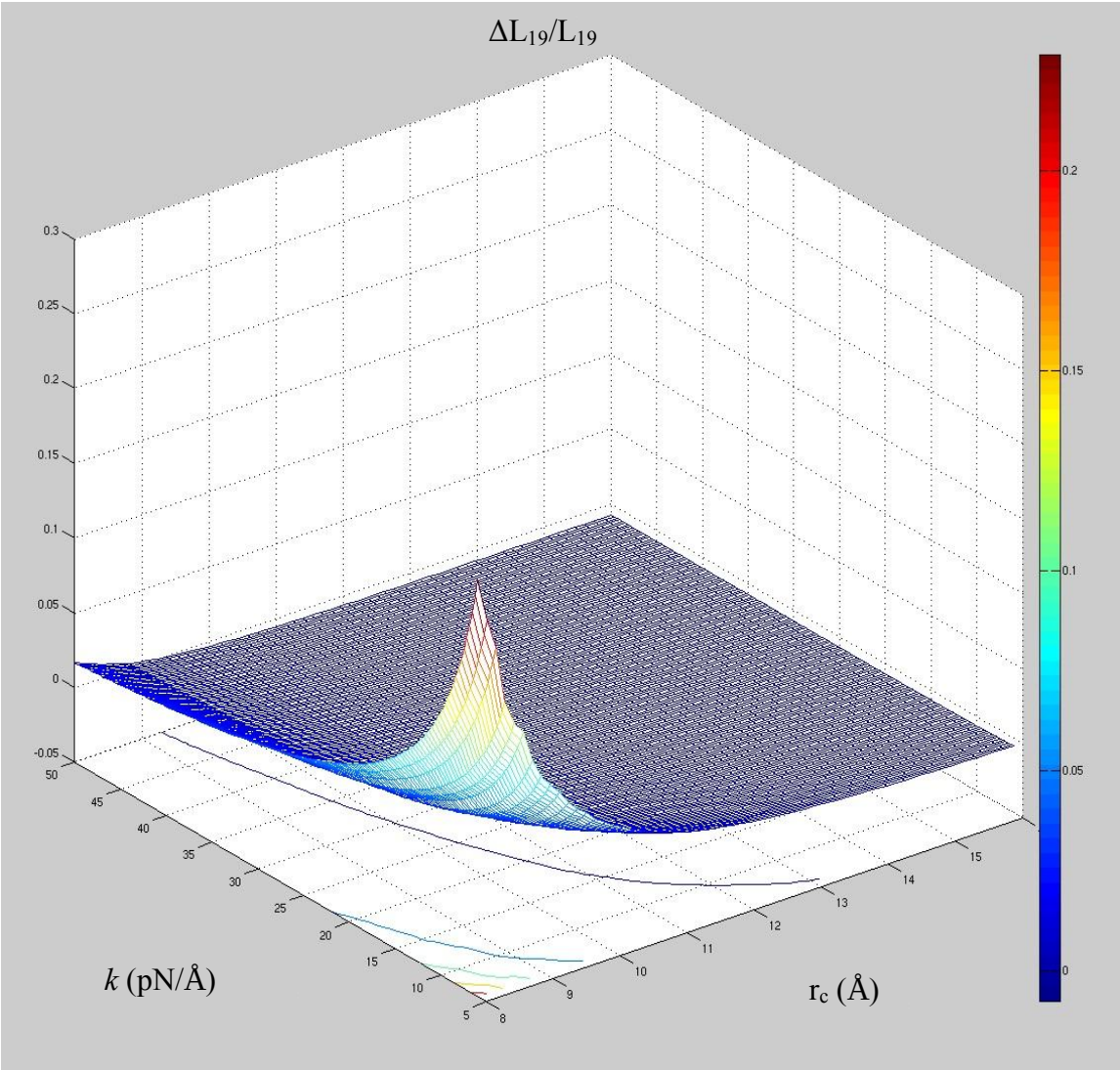


Figure 24. Axial loading r_c versus *k* plane for the $\Delta L_{19}/L_{19}$ metric.

Examination of Figure 24 shows that the SENM model poorly matches the SMD results in the region of low r_c and *k*, denoted by the high $\Delta L_{19}/L_{19}$ value of 0.15. From there, as r_c and *k* increase in size, the $\Delta L_{19}/L_{19}$ metric approaches an ideal match, denoted by a $\Delta L_{19}/L_{19}$ value of 0; from there, the plane levels off around the vicinity of $\Delta L_{19}/L_{19}$ = -0.0075. Just by looking at the figure though, it is difficult to determine where the

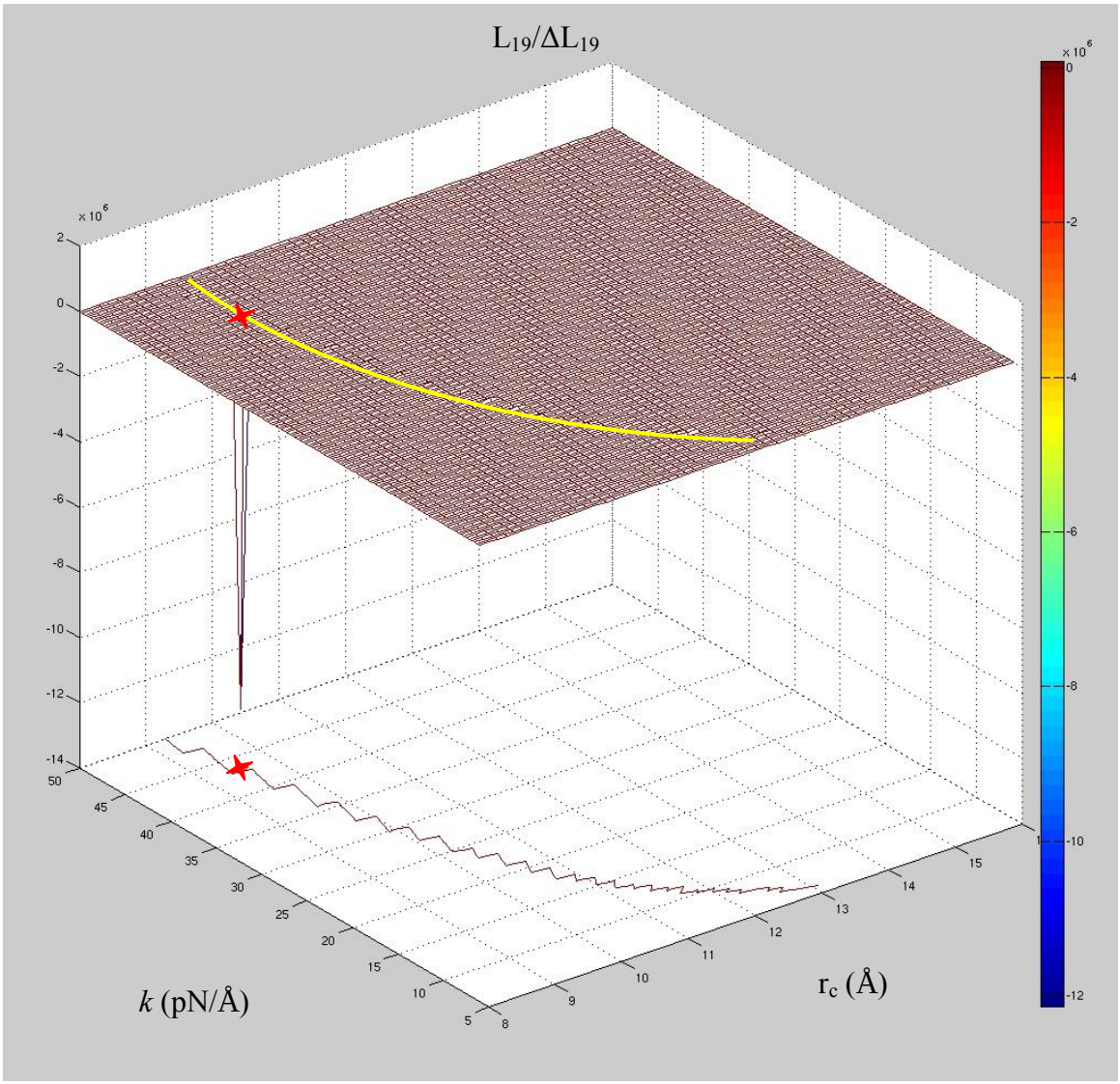


Figure 25. Axial loading r_c versus k plane for the inverse, or $L_{19}/\Delta L_{19}$ metric. Note how values that are optimal stand out as distinct, discrete points. The red star denotes the optimum combination of r_c and k, and the hyperbolic yellow curve follows the surface where the r_c and k combinations are close to optimal.

specific optimum r_c and k combinations occur; therefore, taking the inverse of this plot (where the metric now becomes $L_{19}/\Delta L_{19}$) yields a much clearer indication of where the parameters reach their optimum values (Figure 25). Here, it can clearly be seen that the

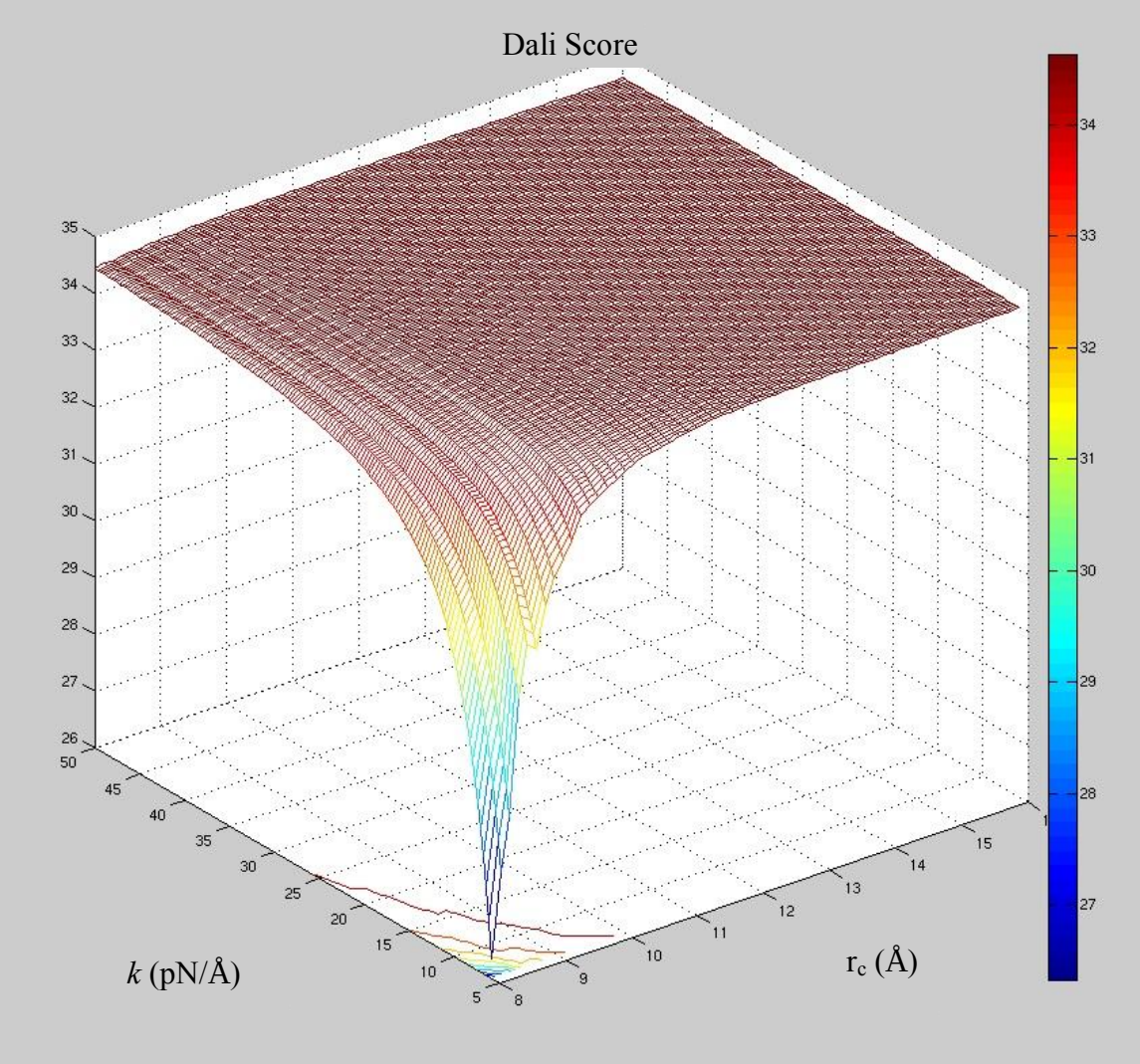


Figure 26. Dali score surface plot of r_c versus k under axial loading conditions.

optimum r_c and k combination occurs when r_c = 9.49 Å and k = 43.3 pN/Å (denoted by a sharp peak in the surface plot).

Looking at the Dali score in Figure 26 shows a similar trend to that seen in Figure 24 in that for low r_c and k values, the SENM model fails to capture the displacements from the SMD results. However, as r_c and k increase, the Dali score does not show any

appreciable increase in the structural match between the two models. In fact, the difference of the Dali scores between the optimum r_c and k values from the $L_{19}/\Delta L_{19}$ residual and that of the highest Dali score on the surface is less that 0.13%. Therefore, because the number of elements in the SENM model increase exponentially with r_c, choosing the added model complexity associated with a higher r_c value cannot be justified.

The results of the residual calculations based on the QoIs prove inconclusive in delivering an optimal r_c and k. As seen in Figure 27 for the $\Delta B_1/B_1$ metric, the general trend of a deteriorating residual at low values of r_c and k still exists, however, a smooth surface does not emerge as r_c and k grow larger as was noticed in the previous metrics. For large r_c and k values, the surface appears flat and stable in that region as $\Delta B_1/B_1$ approaches 0.215, but for increasing r_c and small k values, $\Delta B_1/B_1$ oscillates from a large local minimum to a large local maximum. It is possible that the range of r_c for this metric is not large enough to yield a stable local minimum, but decreasing r_c below 8.0 Å causes a rapid decline in the SENM model residual match while increasing r_c beyond 16.0 Å causes the size of the computational SENM model to grow very large. Therefore, extracting an optimum r_c and k from this figure proves difficult and rather arbitrary. One explanation for this observation is because this metric is trying to quantify the match between the SMD and SENM models by comparing the distance between 2 individual C_α's, which does not appear to be a large enough sample size in order to effectively do so.

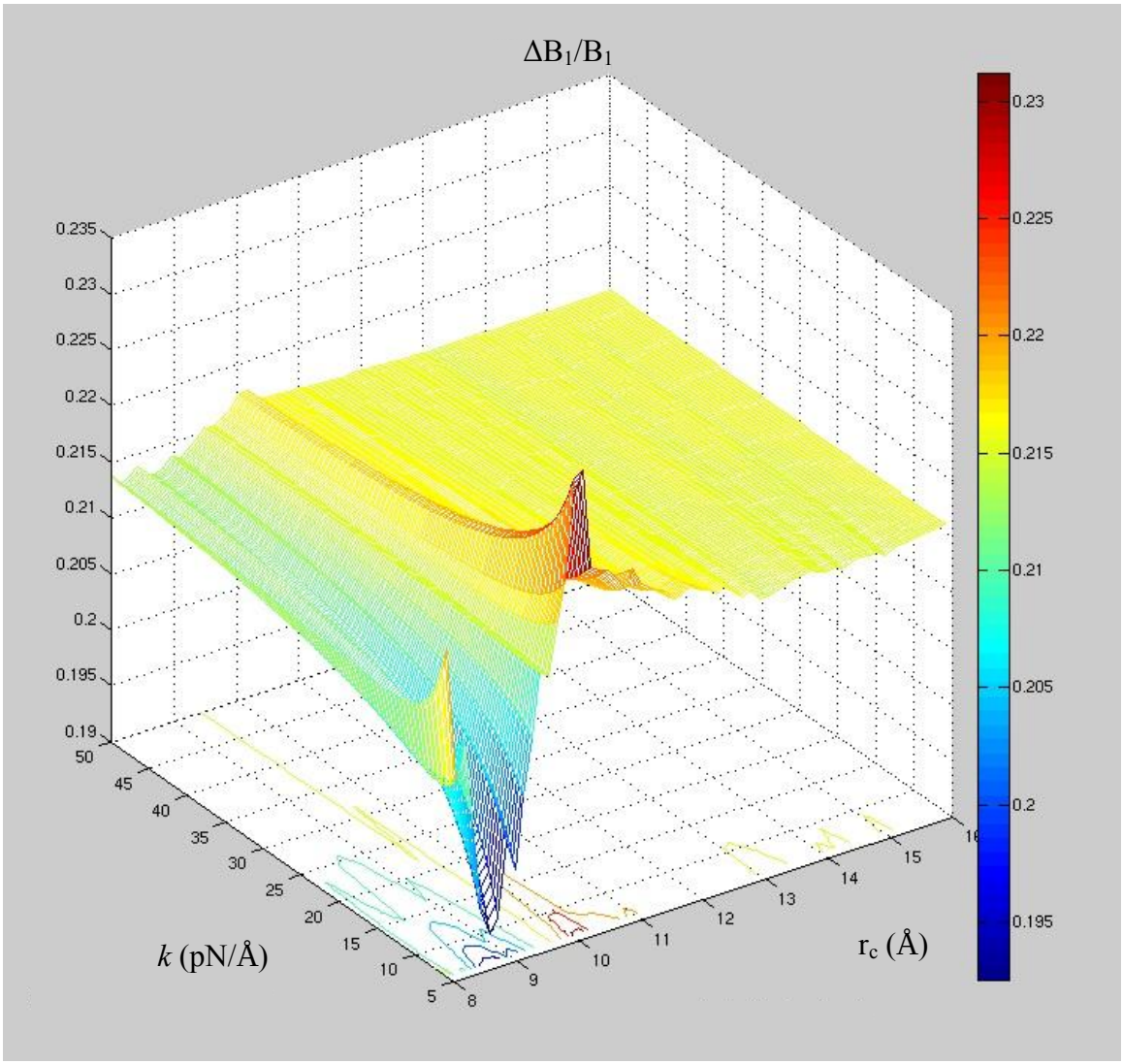


Figure 27. Axial loading r_c versus *k* plane for the $\Delta B_1/B_1$ metric. Notice how the surface adopts a very irregular and jagged terrain.

The results for the $\Delta B_2/B_2$ metric seem to follow a similar trend to the $\Delta B_1/B_1$ metric in a sense that for low *k* values, the surface oscillates between relative local minimums and maximums. The model also tends to degenerate as r_c and *k* approach small values. The similarity between this metric and $\Delta B_1/B_1$ is most likely due to the fact that $\Delta B_2/B_2$ also measures the distance between 2 individual C_α's, which appears very

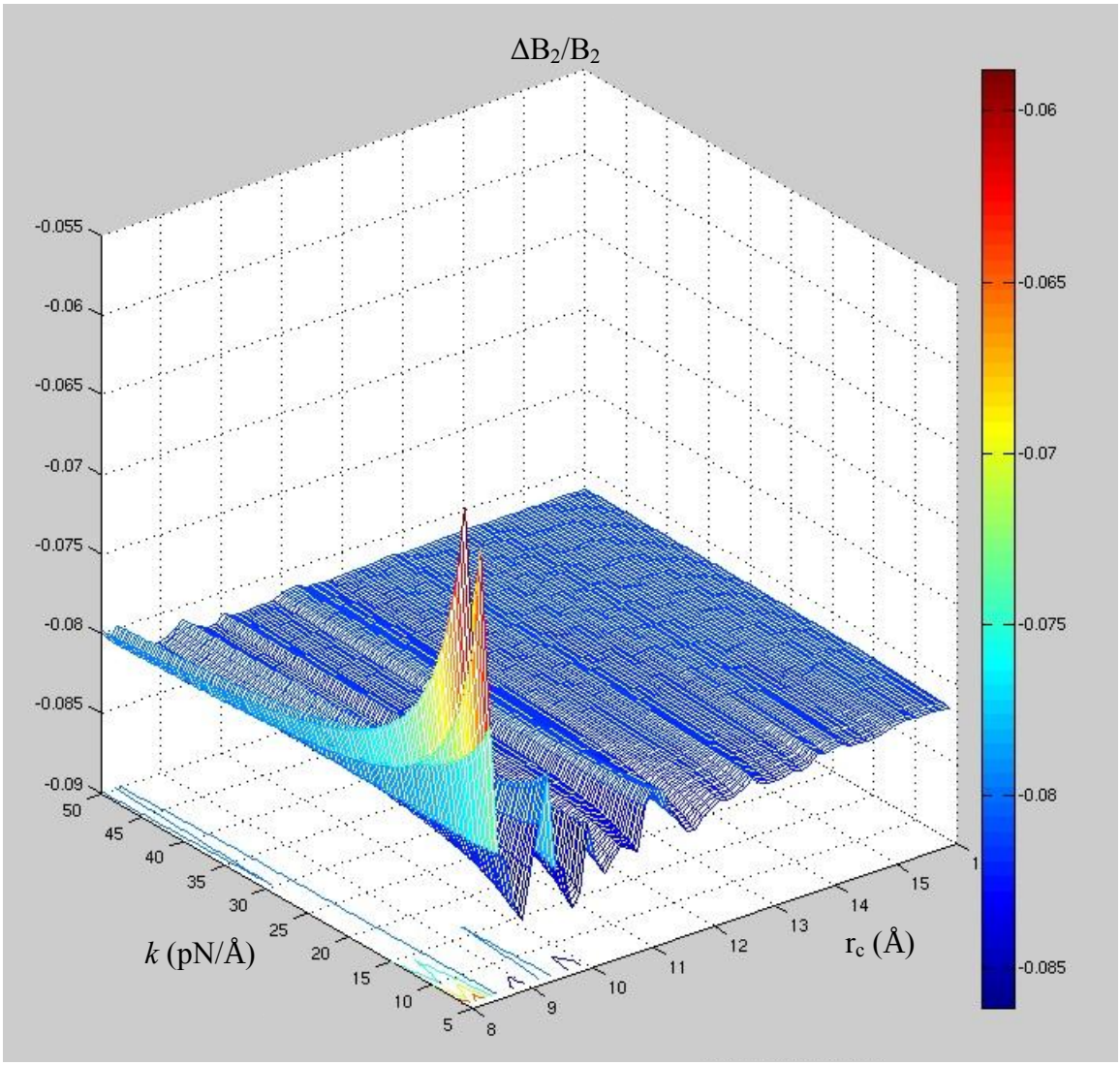


Figure 28. Axial loading r_c versus k plane for the $\Delta B_2/B_2$ metric. The surface fluctuates significantly between relative high and low spots as r_c increases.

sensitive to the stochastic nature of MD simulations (this topic will be discussed in further detail in the subsequent sections).

The $\Delta D_{inter}/D_{inter}$ metric generated a surface plot that is very smooth, and approaches to a local minimum when r_c is close to 8.0 Å (Figure 29). The issue that

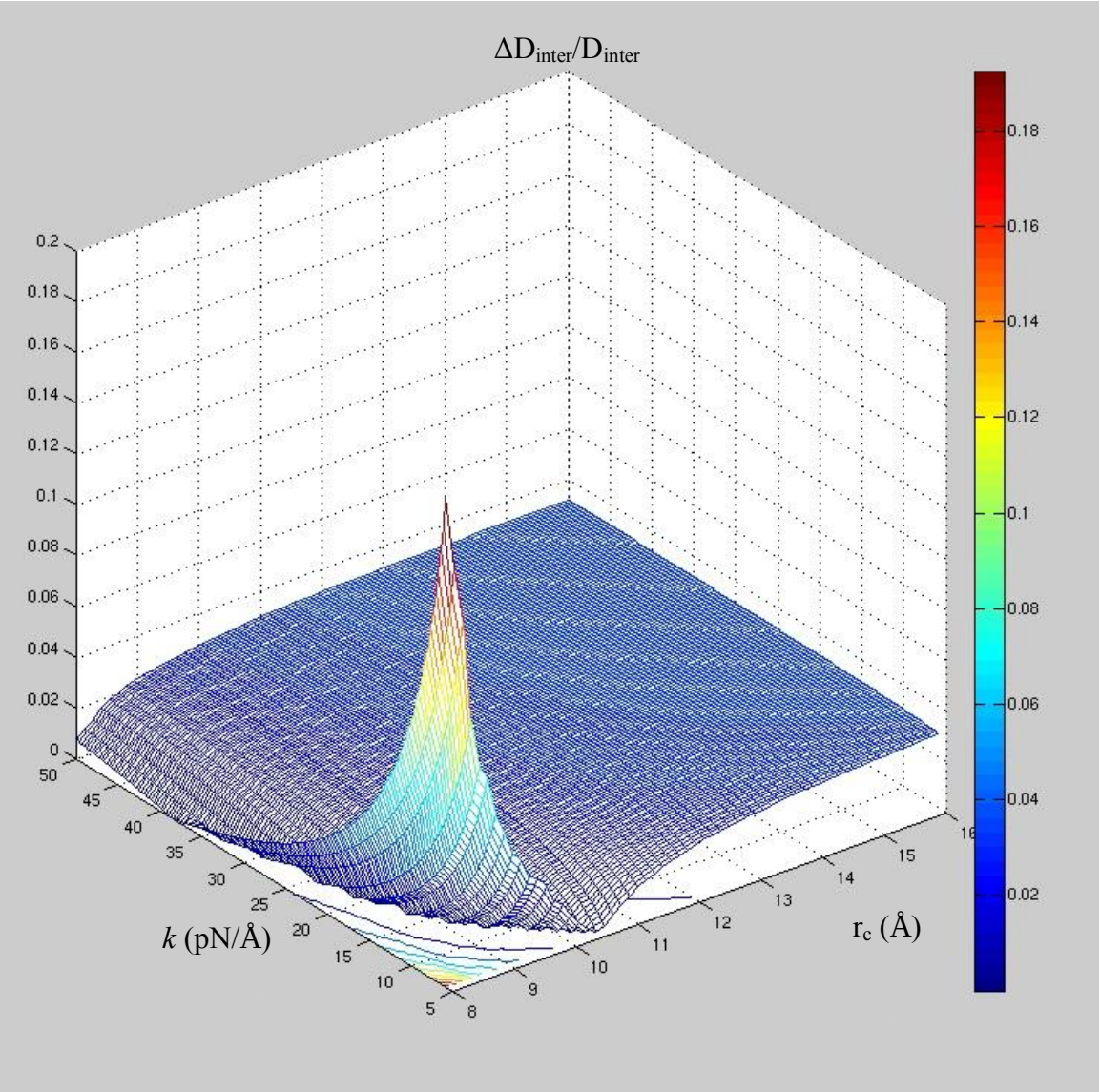


Figure 29. Axial loading r_c versus k plane for the $\Delta D_{inter}/D_{inter}$ metric. The surface appears quite smooth and reaches a local minimum at very low values of r_c and k.

arises with this metric is the fact that the residual is still falling at 8.0 Å, implying that the metric has not yet converged. As mentioned earlier, if r_c drops too low, especially below 8.0 Å, the model tends to lose its match to SMD results, especially with respect to other

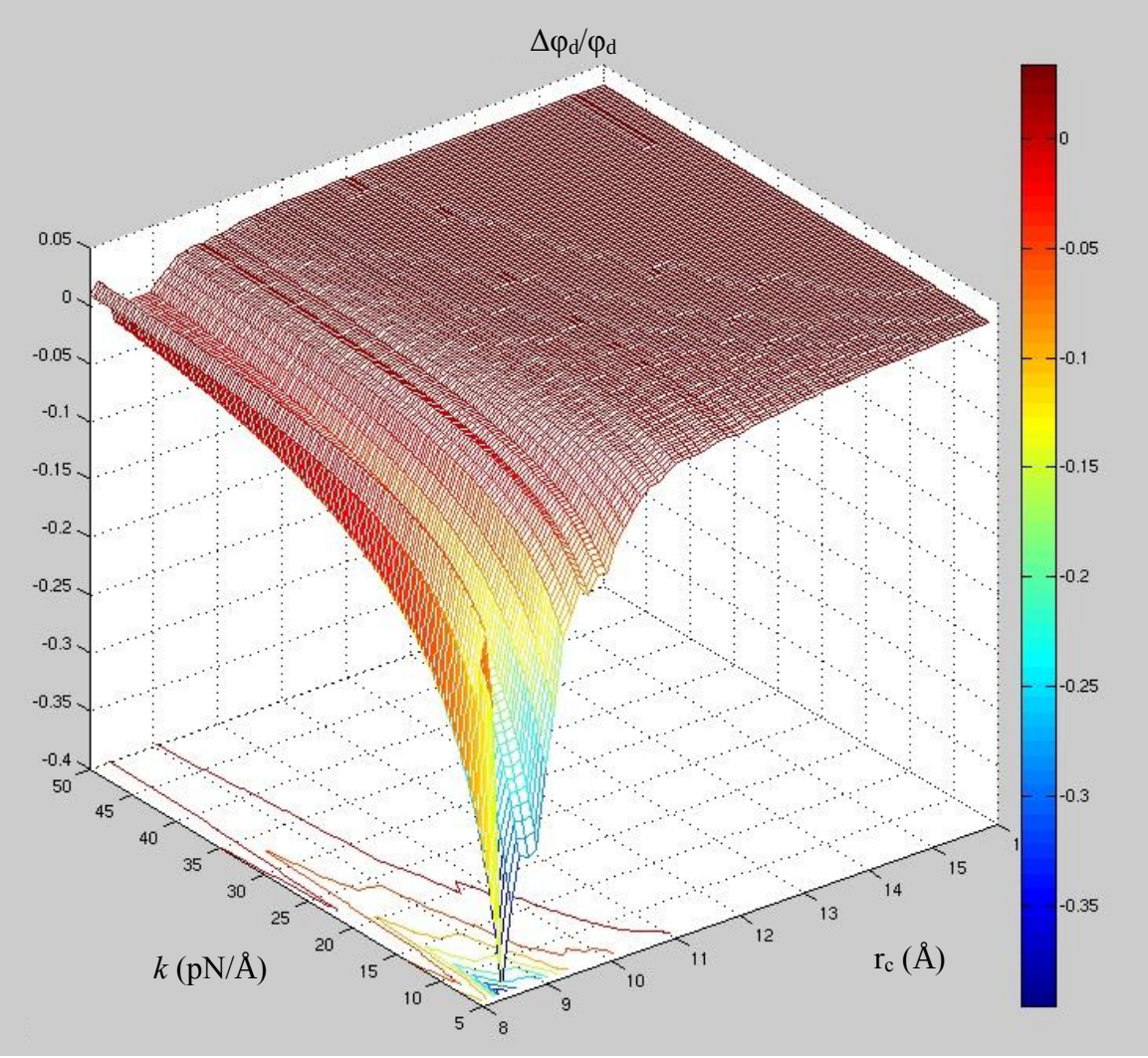


Figure 30. Axial loading r_c versus *k* plane for the $\Delta\varphi_d/\varphi_d$ metric. Notice how the surface apprears smooth at larger values of r_c but tends to become very irregular at r_c values less than 10.0 Å.

metrics, such as the Dali score and the $\Delta L_{19}/L_{19}$ distance. The surface smoothness observed with the $\Delta D_{inter}/D_{inter}$ metric is likely due to the fact that the distance over which this metric spans, which is approximately 5.5 nm (the length of a g-actin monomer), is much larger in magnitude than the B_1 and B_2 distances, which proved very sensitive to MD fluctuations.

The last metric, which compares the dihedral angles between the SMD and SENM models, presents an issue in matching the two models for r_c in the range between 8.0 and 10.0 Å. As seen in Figure 30, at an r_c of 8.0 Å the surface starts off with a positive $\Delta\varphi_d/\varphi_d$ value, then crosses 0, goes negative, and then returns positive as r_c increases. This implies that because the residual surface crosses 0 in numerous locations, there are many values that correspond to a local minimum where the SMD and SENM models match. Therefore, this metric proves unreliable and would require additional processing to determine which r_c and *k* parameters are optimal. Table 3 summarizes the parameterization results for the SENM structure under axial load. The results from the $L_{19}/\Delta L_{19}$ metric will be used in the subsequent section as the optimal SENM r_c and *k* values.

Optimal Parameters for SENM Model		
Metric	**rc**	**k**
$L_{19}/\Delta L_{19}$	9.49	43.3
Dali Score	>9.0	>20.0
$\Delta B_1/B_1$	inconclusive	inconclusive
$\Delta B_2/B_2$	inconclusive	inconclusive
$\Delta D_{inter}/D_{inter}$	8.0	38.5
$\Delta\varphi_d/\varphi_d$	multiple	multiple

Table 3. Summary of SENM model parameterization under axial loading. The results from the $L_{19}/\Delta L_{19}$ metric are used in the final model. The Dali score remained relatively the same for values larger than the ones shown. The remaining four metrics were either inconclusive, or gave an r_c and *k* combination that was too low for the model.

4.3.3 SENM Axial Results

Upon arriving at the optimum r_c and *k* parameters for the SENM model, specific model properties and comparisons can now be extracted from the results. Figure 31

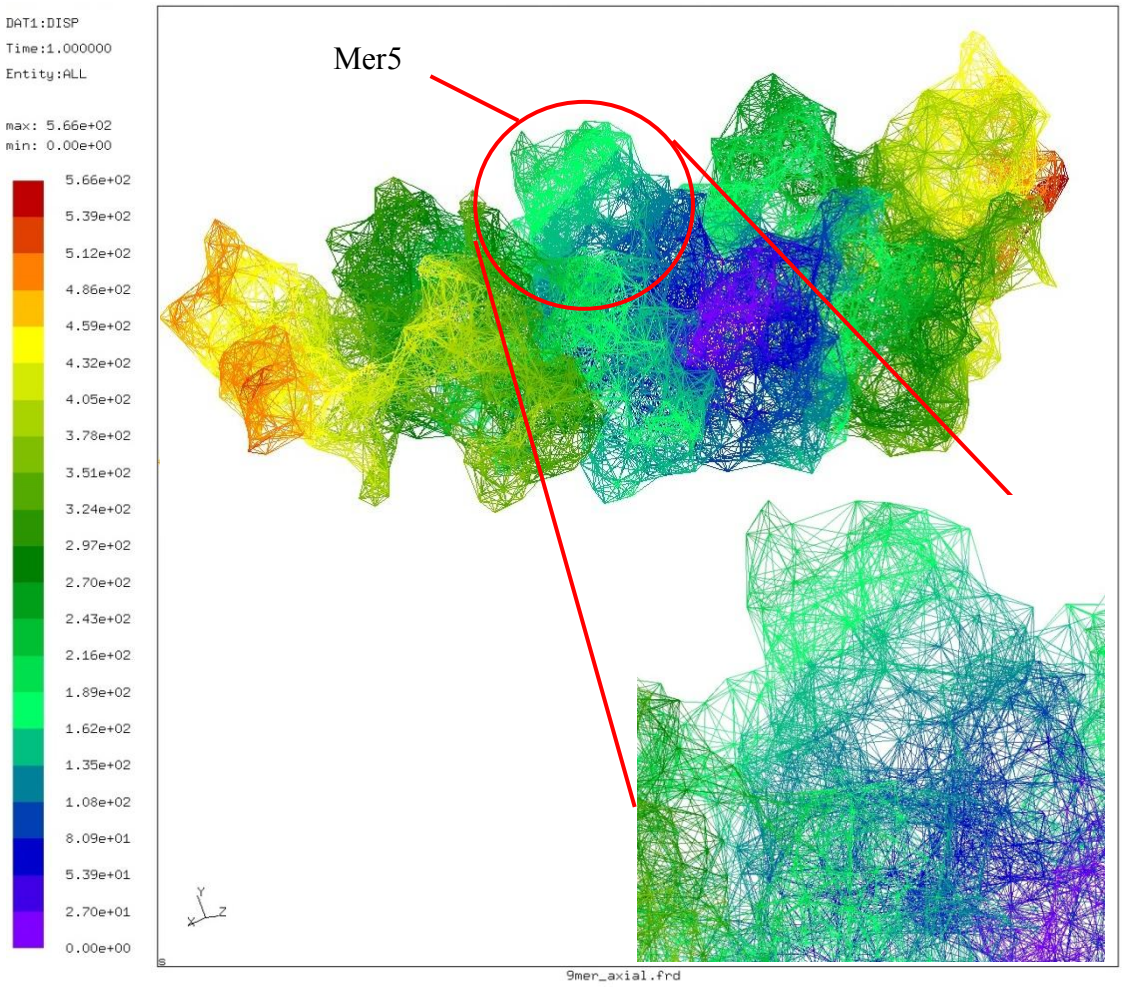


Figure 31. Screen shot of the entire axially loaded 9mer structure from CalculiX's visualization program GraphiX. While individual monomers are difficult to differentiate due to the unbiased assigning of elements between filament nodes, mer5 has been encircled to show its relative position in the filament network.

shows an example screen shot of the output from *Calculix's Graphix* visualization tool. It also shows the levels of complexity in the network by how densely packed the elements appear. For example, in a 9mer filament network with an r_c of 9.49, there are 29,221 elements connecting all 3,375 nodes, which averages around 8-9 elements per each node. Due to the lack of clarity seen with *GraphiX*, PyMOL will be used for the remaining protein displacement images.

While according to the $L_{19}/\Delta L_{19}$ residual that says the SENM and SMD simulations match up well based on the overall stiffness metric, the individual residue displacements contradict this notion. One possible explanation for this discrepancy is that while the CoM (which is an *average* of the individual residue positions) of mers 1 and 9 displace the same distance as in the SENM model, the side-by-side comparison of mer5 residue positions for the SMD and SENM structures shows how strong the influence of random fluctuations inherent to the SMD simulations are (Figure 32).

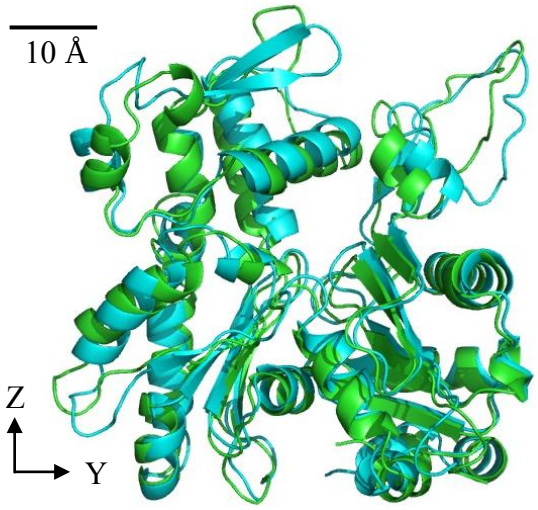


Figure 32. Mer5 from the SMD (cyan) and SENM (green) structures are overlaid on each other to show that while the CoM of each structure's mer5 match up, the individual residue positions do not. Secondary structures such as α-helices and β-sheets are approximately in the same position between the two structures, but in the SMD simulations, less stiff regions such as loops are very sensitive to thermal fluctuations (figure generated in PyMOL).

As seen in Figure 33, the magnitudes of the displacements for the SENM model are much smaller than observed in the SMD results. For mer5, the RMSD of the displacements are on the order of 0.107 Å, an order-of-magnitude less than the SMD RMSD of 2.78 Å. However, if you look at the locations of where the large displacements

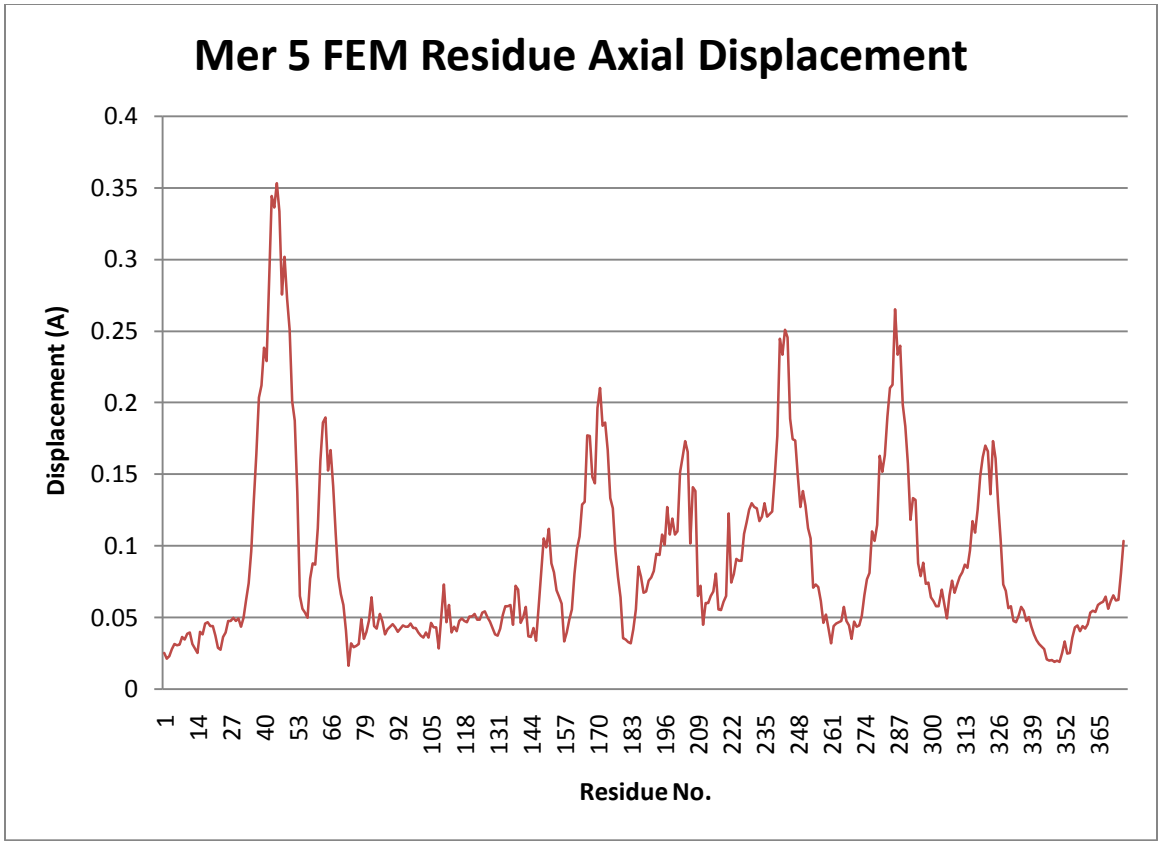


Figure 33. Mer5 magnitude of SENM residue displacements for an applied axial load of 200 pN. The DB-loop around residues 40-48 undergoes the largest displacements, upwards of 0.35 Å. Other key areas of large displacement include the loop around residues 240-248 and 322-325. The overall structure RMSD is 0.107 Å.

occur, they tend to be in the same regions as seen in the SMD simulations. For example, in the SENM model, the DB-loop undergoes some of the largest displacements of the whole monomer, up to 0.35 Å, in addition to the loop encompassed by residues 240-248. The loops around residues 287 and 322-325 also undergo large displacements of 0.25 Å and 0.17 Å, respectively.

As was noticed in the SMD simulations, majority of these residues are located on the outer regions of the monomer, specifically in areas that are known to bind to

neighboring residues (Figure 34). In addition, the recurring theme that regions containing a loop morphology with few hydrogen bonds to stabilize the structure leads to the conclusion that these areas will undergo some of the largest displacements in that monomer. In addition, Figure 34 shows that areas in direct contact with their neighbors are more likely to deform larger distances than those residues in a more central position of the protein.

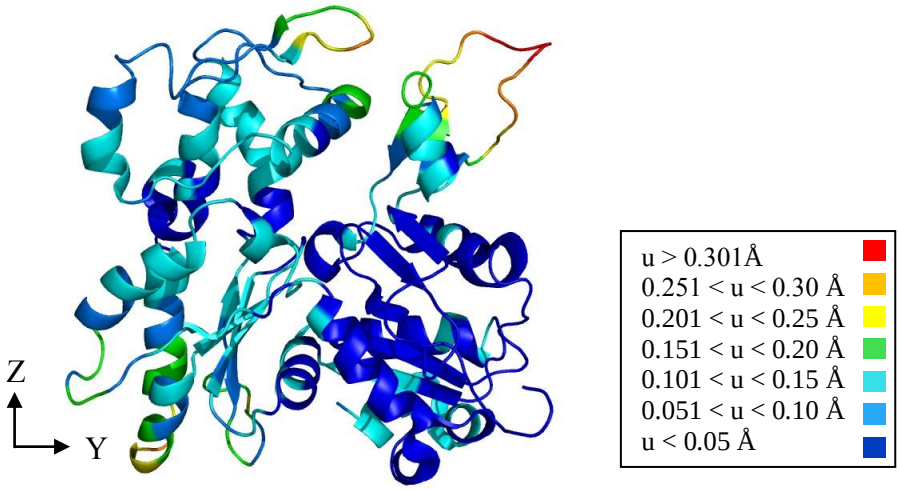


Figure 34. Ribbon representation of the magnitude of residue displacements of the SENM model under 200 pN applied axial load. The DB-loop and the loop around residues 240-248 and 322-325 are some of the areas that undergo the largest monomer displacements (figure generated in PyMOL).

4.4 TORSIONAL LOADING RESULTS

This section now presents and discusses the results from the 200 pN-nm torsional load that has been applied to the 9mer F-actin structure. Both the SMD and SENM model loading results are shown, along with the results of the SENM model parameterization.

4.4.1 SMD Torsional Results

As mentioned in section 4.3.1, the data that is presented and discussed in the following section is from SMD simulations performed by Steven Kreuzer [69]. Subjecting an actin filament to a torsional load yields quite a different displacement field that observed from axial loading. While the RMSD of mer5 in the SMD structure is ~2.767 Å, almost identical to that of the axial structure, there are some key areas that do not move in the same manner.

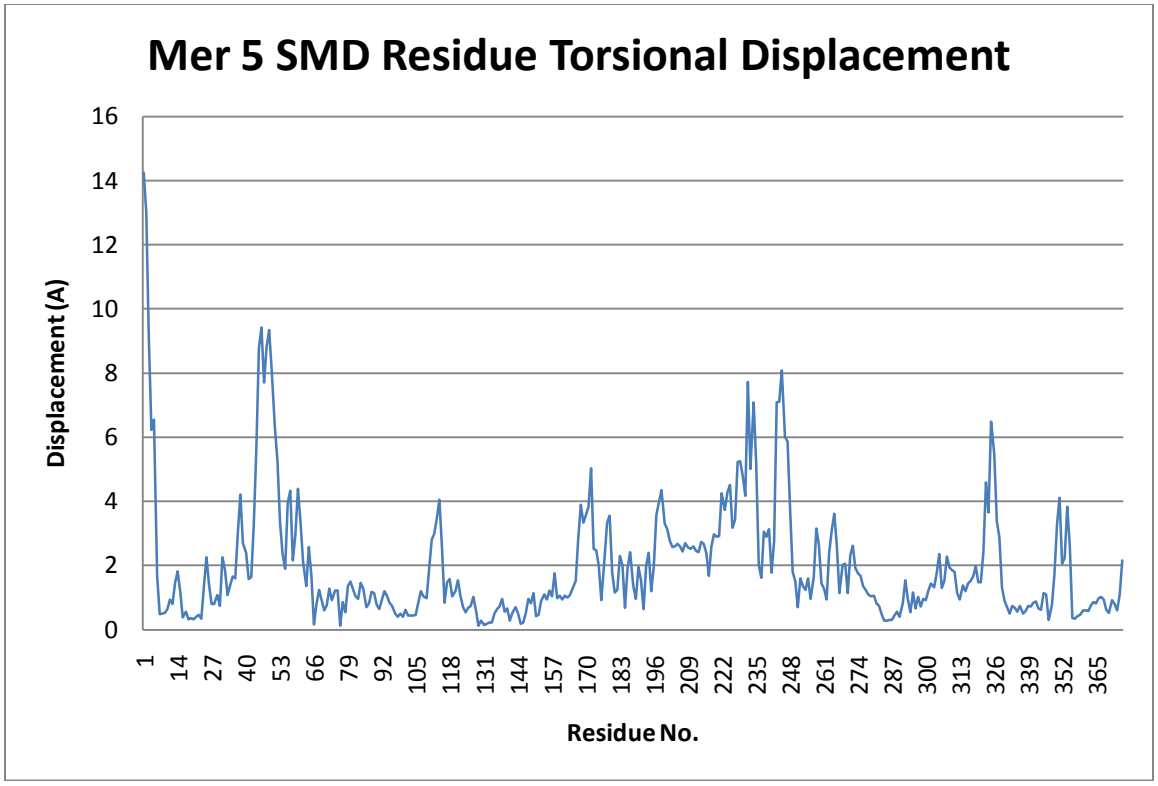


Figure 35. Mer5 SMD residue displacements under a 200 pN-nm torsional load. The non-bonded N-terminus (~residues 1-5) undergoes very large displacements up to 14.0 Å. Regions of relatively large displacements include the DB-loop, the loop around residues 240-248, the α-helix around residues 223-230 and the loop around residues 322-325. Data from [69].

Under torsional loading, the DB-loop is subjected to higher stresses and, because of its weaker loop geometry, has large displacements upwards of 9.0 Å (Figure 35). The loop around residues 110-112, which binds to the monomer that is across the filament coil, also sees an increased displacement up to 4.0 Å. Due to the geometry of this bond, the higher displacement in this region makes intuitive sense because the applied torque will want to 'untwist' the two helical filament coils away from each other (as noticed in the two coils shown in Figure 15). As also observed for the monomer under axial load, the α-helix around residues 223-230 displaces up to 8.0 Å in this simulation (Figure 36). Again, because this region is not known to bind to any surrounding monomers, this is likely due to thermal fluctuations inherent to SMD simulations.

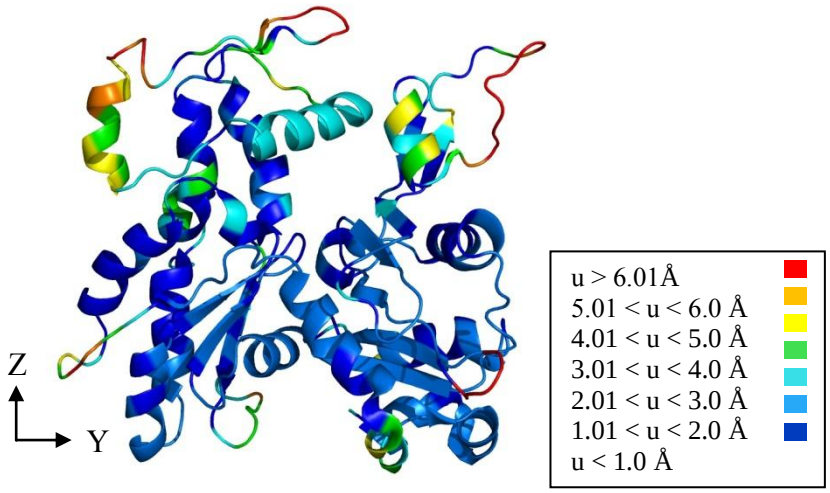


Figure 36. Ribbon representation of mer5 displacements from the SMD simulations under a 200 pN-nm torsional load. Residues from 240-248, the DB loop and the α-helix by 223-230 all undergo displacements larger than 5.0 Å. Data from [69] (figure generated in PyMOL).

4.4.2 Torsional Load Parameterization/Validation

Using the same automation code from Dr. Liu [74], the best residual metric to be used in selecting the optimum r_c and k is the Dali score. A similar parameterization technique as was presented for axial loading in section 4.3.2 is used for torsional loading, the results of which are summarized in Table 4. Looking at the other 4 residual metrics used to evaluate the torsional load, which include $B_1/\Delta B_1$, $B_2/\Delta B_2$, $D_{inter}/\Delta D_{inter}$ and $\varphi_d/\Delta\varphi_d$, the $B_1/\Delta B_1$ and $B_2/\Delta B_2$ residuals tended to generate surface plots that are very similar to the Dali score metric, while the $D_{inter}/\Delta D_{inter}$ and $\varphi_d/\Delta\varphi_d$ metrics proved inconclusive (which explains why the Dali score was used for torsional loading parameterization). Generally, the QoI metrics degraded as r_c and k were small and tended to level off as r_c and k grew in size.

Optimal Parameters for SENM Model		
Metric	**rc**	**k**
Dali Score	>9.0	>20.0
$\Delta B_1/B_1$	>9.0	>20.0
$\Delta B_2/B_2$	>9.0	>20.0
$\Delta D_{inter}/D_{inter}$	inconclusive	inconclusive
$\Delta\varphi_d/\varphi_d$	inconclusive	inconclusive

Table 4. Summary of SENM model parameterization under torsional loading. A combination of the results from the Dali score and the optimum parameters from the axial loading case are used in the torsional loading scenario.

As can be seen in Figure 37, the evolving residual surface looks quite similar to the surface from the axial SENM parameterization. As r_c and k values approach their lower limits, the Dali score quickly drops below 33.6, indicating a poor match to SMD

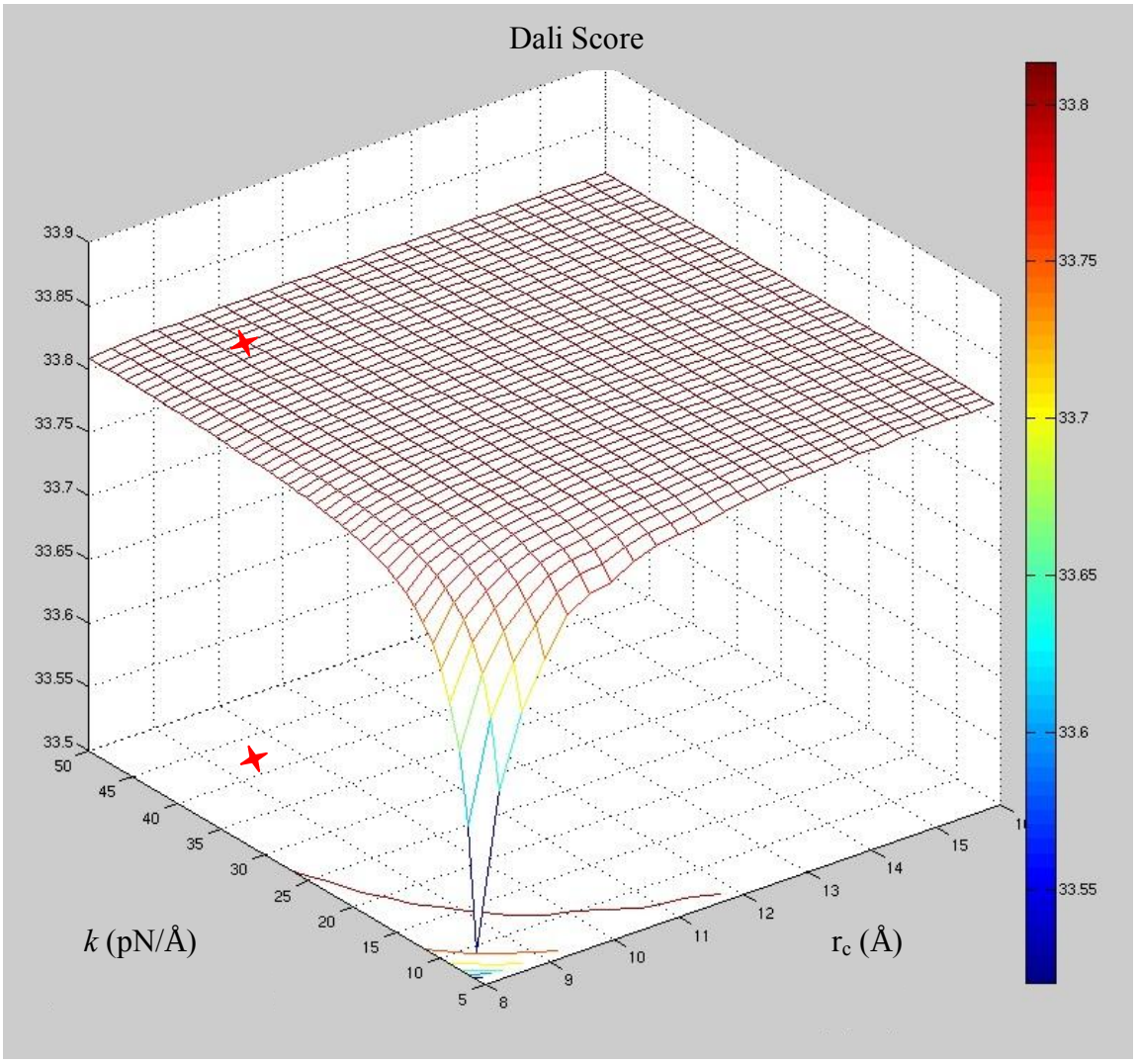


Figure 37. SENM Dali score surface plot of r_c versus *k* under 200 pN-nm torsional load. The optimal r_c and *k* parameters used in the axial loading case is marked on the surface with a red cross.

results. In addition, at larger r_c and *k* combinations, the Dali score quickly levels off past 33.8 as the slope of the surface approaches 0. Using the same r_c and *k* parameters that were obtained in the axial loading parameterization yields a Dali score of 33.8118. Furthermore, the highest calculated Dali score on the torsional r_c versus *k* surface was 33.8132; while this point corresponds to the optimum r_c and *k* values for torsional

loading, the Dali score is only different by 0.004%. Therefore, for the sake of decreasing computational expense, the lower r_c and k values obtained from the axial parameterization will be used in the torsional SENM simulations.

4.4.3 SENM Torsion Results

The displacements observed in mer5 under torsional loading have much larger displacements, and a markedly different displacement field than its axial counterpart. The RMSD value of the displacements is 0.670 Å, which shows that the monomer is much

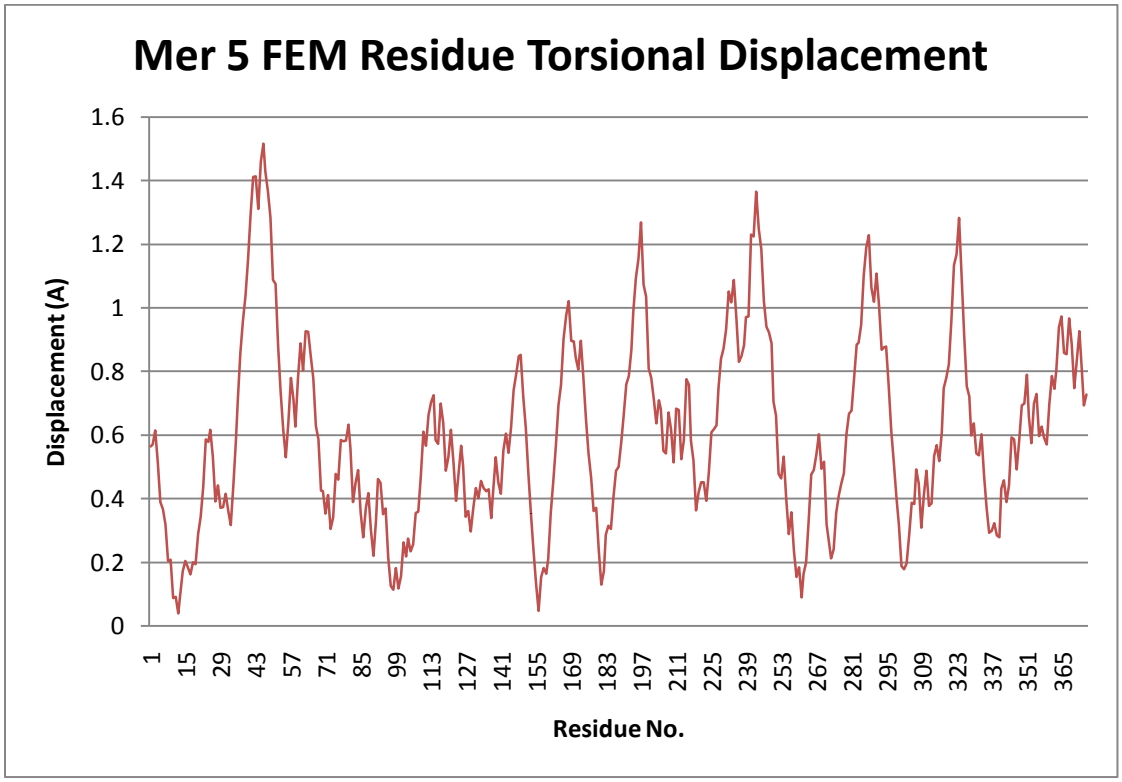


Figure 38. Mer5 SENM residue displacements for the 200 pN-nm torque load. Larger SENM displacements are observed, with mer5 having an RMSD of 0.670 Å. Protein regions of higher displacement correspond to areas on G-actin that are on the perimeter of the structure, including residues 40-48 (DB-loop), 166-170, 194-204, 239-247, 285-290, and 320-325.

more flexible when subjected to a torque load as this is not the main mode of loading F-actin as observed *in vivo*.

As can be seen in the plot of residue displacements in Figure 38and as graphically displayed in Figure 39, more regions of the monomer are recruited and displaced under a torsional load. Essentially all of the residues on both the top and bottom of mer5 (residues 40-48 (DB-loop), 166-170, 194-204, 239-247, 285-290, and 320-325) undergo displacements that are greater than 0.80 Å. The effect of torque on the monomer is causing relative motion between the subdomains, where SD2 and 4 are moving to the left and SD 1 and 3 are moving towards the right. This is why the middle portion of the monomer has relatively no displacements compared to the other parts of the protein. This motion of mer5 is expected though because the 200 pN-nm torque was applied in a manner that would tend to unravel the actin as a double helical polymer. The effect of

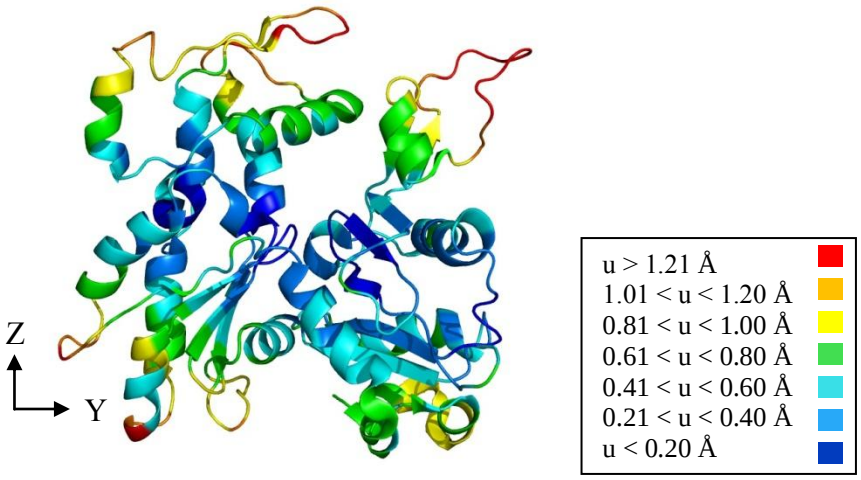


Figure 39. Ribbon representation of SENM model residue displacements under 200 pN-nm torsional load. Regions along the perimeter of G-actin tend to have the largest displacements. Specifically, residues 285-290 displace 1.2 Å, which is the largest displacement this region has seen compared to all of the previously mentioned loading conditions and simulations (figure generated in PyMOL).

torque on the structure is discussed in more detail in the follow section in the B1, B2 and φ_d QoI sections.

4.4.4 QoI summary table

In addition to the overall mer5 results previously presented for axial and torsional loading, there are four addition QoIs that are to be investigated to see how well the SENM model can match the residue level displacements of the SMD simulations. Table 5 shows the SMD mean and standard deviation (σ) results [69] next to the mean results

QoI	Loading Scenario	SMD [69]		SENM-FEM
		Mean	σ	Mean
B1 (Å)	Eqm	5.009	0.3	5.009
	Axial	4.122	0.2	5.007
	Torsion	4.222	0.2	5.029
B2 (Å)	Eqm	7.662	0.4	7.662
	Axial	8.34	0.3	7.659
	Torsion	8.085	0.3	7.717
Intermer Distance (Å)	Eqm	53.118	0.7	53.118
	Axial	54.883	0.7	54.086
	Torsion	52.596	0.9	55.136
Dihedral Angle (°)	Eqm	8.586	2.53	8.586
	Axial	8.302	2.44	8.434
	Torsion	12.321	3.07	7.184

Table 5. QoI table summarizing and comparing the results of the SMD and SENM simulations. B1, B2 and intermer distances are in angstroms (Å) while the dihedral angle is in degrees (°). SMD data from [69].

from the SENM FEM implementation for B1, B2, the intermer distance and the monomer dihedral angle.

First of all, it should be noted that the SMD results all have a σ associated with every QoI while the SENM results do not. This is due to a number of reasons. First of all, the SENM implementation is deterministic in a sense that upon loading, the structure will deform and remain in that position. This is not to say that there is no variance or error associated with the SENM model, because clearly this is not true. In fact, if the displacement results of the FEM simulations were presented for a range of r_c and k, in a similar manner to that of determining the optimum r_c and k values, then a variance, or error range could be provided along with the presented mean values. Instead, the SENM results are presented as deterministic (because r_c and k values have already been optimized), and are compared to the SMD results in the context of the given σ's.

A recurring theme observed with the SMD results are the large RMSD values associated with residue displacements. While the structure used in the analysis is an average of the final 2.5 ns in the simulation, the mean positions of individual residues are still moving distances between 2.0 and 3.0 Å, over an order-of-magnitude larger than the displacements observed in the SENM simulations. This could be due to the fact that within the 10 ns simulation time, the SMD structure is still evolving and redistributing the applied load through the structure. If this is the case, an average over an evolving system will not yield confidence in the final structure. It is also possible that as mechanical loads are applied to the SMD structure, the residues are constantly rearranging themselves between areas of low potential energy. This could cause subtle

conformational changes to occur throughout the protein, and disrupt any notion of a final, stabilized deformed structure.

The SENM QoIs presented in Table 5 all match the SMD results to within 1-σ with the exception of B1. This comes as a surprise though because B2 is essentially a measure of the same quantity, merely one residue away from B1 in the NBP, yet the SENM B2 value matched the SMD results well.

In spite of this information, the B2 values in both loading scenarios remain >7.0 Å, implying that the deformations are not large enough to cause any change in the NBP state. However, an examination of the equilibrium structure's B2 distance of 7.6 ± 0.4 Å shows that the NBP already started out in the open position, which is not energetically favorable a stable actin filament. Taking this into account means that had the equilibrium B2 value been smaller (i.e. closer to 6.0 Å), it could be possible that at maximum physiologic load the NBP could change from a closed to intermediate or open state.

Axial and torsional loading does increase the SMD B2 from 7.6 ± 0.4 Å to 8.3 ± 0.3 Å and 8.09 ± 0.3 Å, respectively while SENM results show that B2 slightly decreases for axial loading and slightly increases under torsional load. For this particular QoI, it is difficult to decipher which direction the SENM results want to move the structure, while it is obvious from the SMD results that B2 should increase under load. It does appear that residues 14 and 158 (B1) are getting closer to each other by ~1.1 ± 0.2 Å in both axial and torsional loading, according to the SMD simulations. The SENM model also indicates a closing of the NPB for axial loading, yet B1 increases for torsion.

The intermer distance under axial load between SMD and SENM simulations match very well, but the torsion simulations do not. The SENM model appears to capture the residue level displacement from axial loading to within 0.07 Å even though the displacement itself is more than 1.0 Å. Under torsional load, the SENM model displaces around 2.036 Å, which for the SENM simulations is a very high value. It seems that the intermer bonds between residues on the edge of two monomers deform more because of the increased initial distances between residues along the boundary of two monomers. This boundary condition causes fewer elements to be assigned to the residues in the region during initial network creation, therefore implying that the SENM model does not seem to work well within one particular monomer, but has the strength to emulate inter monomer bonds.

The SMD dihedral angle for axial loading changed very little from the equilibrium value of 8.59 ± 2.53° to 8.30 ± 2.44°, and came very close to matching the SENM value of 8.43°. Therefore, the application of a maximum physiological axial load has a very small effect on the dihedral angle of a G-actin.

When F-actin is subjected to a torsional load, the SMD simulation results in a large φ_d increase to 12.3 ± 3.07°, while the SENM model actually caused a decrease in φ_d to 7.2°. The large increase in φ_d can be seen in Figure 40. Because of the geometry of G-actin, it can be inferred that under torsional loading, an increase in φ_d coincides with an opening up of the NBP, which is exactly what was observed in the SMD simulations.

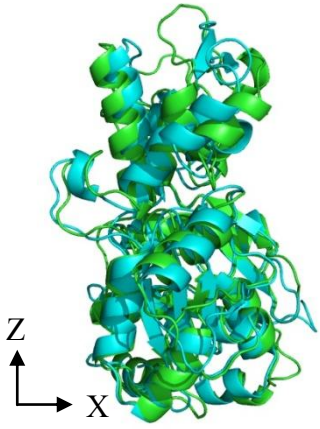


Figure 40. Side view of mer5 under torsional load illustrating the large change in the dihedral angle, φ_d, from 8.59 ± 2.53° to 12.3 ± 3.07°. The equilibrium structure is in green and the deformed SMD structure is highlighted in cyan (figure generated in PyMOL).

4.5 UNLOADED VERSUS LOADED PARAMETERIZATION COMPARISON

One of the main motivating reasons for decoupling r_c and *k* was to investigate the effect that loading has on the parameterization of the ENM. From the initial unloaded 9mer normal mode analysis, the optimum r_c (and corresponding *k* of 17.9 pN/Å) occurred at 12.5 Å. After axial loading parameterization though, an optimum r_c of 9.49 Å (and corresponding *k* of 43.3 pN/ Å) was calculated, which is quite lower than the unloaded results (especially because the number of elements between those two r_c's almost doubles from 29,221 (r_c of 9.49 Å) to 55,584 (r_c of 12.0 Å)).

The mismatch between the unloaded and loaded F-actin parameters most likely stems from the observation that an unloaded 9mer filament undergoes a sinusoidal bending regime along its length as the first mode of oscillation. Therefore, the unloaded model requires additional stiffness along that loading direction while for axial and

torsional loading, the loading directions require different nodal connectivity. Had the SENM model been subjected to a bending scenario, it is very likely that the unloaded and loaded parameters would have matched up well.

If the SENM model was parameterized solely on the results from the unloaded normal mode analysis, the Dali score and output displacements would be insensitive to the higher r_c parameter. However, using the Dali score and $L_{19}/\Delta L_{19}$ metric to further parameterize the model has allowed us to optimize the r_c and *k* values to create a network that is simpler and more computationally efficient (fewer number of elements) from what was otherwise thought optimal. It is possible that the observed increased structure flexibility in the torsional model is a result of parameterization based on the optimal axial results.

4.6 MODEL IMPLICATIONS AND DISCUSSION

Using the ENM to quantify the displacements of residues in a protein due to external loads has proved a difficult and involved process. In addition, there are many assumptions and simplifications made in order to take such a complex protein and simplify it to a two parameter model system. That being said, there are still a number of things to take away from these results.

First off, the nature of the SMD simulations show that the thermal fluctuations present in these simulations cause a great deal of localized disorder shown by the RMSD value of the displacements. Furthermore, the range of displacements from the axial and torsional SMD results clearly show that it is possible for some residues to move up to 14.0 Å, almost a quarter the diameter across an actin monomer. It is also possible that

under load, proteins undergo subtle conformation changes that can lead to unpredictable displacements and rearrangements in the protein structure. In fact, similar RMSD values around 2.1 Å have been observed in unloaded simulations of the G-actin monomer, further illustrating the stochastic nature of MD simulations [23]. This information implies that regardless of what ENM one may choose to use in quantifying a protein's response to a structural load, the bottom line is that the SMD results are going to have a lot of variation, and possible associated error with the results. Therefore, having σ, the associated error range of the SMD results, greatly helps in quantifying the statistical significance between the SMD and SENM model results.

Conversely, the SENM, which, by means of the network's mesh definition, generates a continuum type network which tends to smooth out any areas of highly localized deformation. Within the context of F-actin though, it is difficult to say whether this observation is due to the low magnitude of applied external loads (which are on the same order of magnitude as those maximally applied *in vivo*) or if it is a property inherent in the ENM. It is possible though that because of the ENM's potential definition, the SENM will only be able to describe the local fluctuation of a protein near the equilibrium structure. While the ends of the SENM filament are subject to the same loads as the SMD simulations, the observed displacements in mer5 of the SENM are generally an order-of-magnitude lower (as calculated by the RMSD values in the two models). In this regard, the SENM does not seem to correspond well to the SMD displacements. However, the specific regions that undergo larger displacements in the two models seem to match each other quite well. Therefore, it appears as if the SENM is capturing the essence of how regions in mer5 are deforming under load, but due to its simplicity is missing another factor at play.

Building on this thought, it is apparent from the axial simulations (and from available *in vitro* experiments [17]) that the actin filament is very stiff under axial load. As an example, a 9mer structure around 25.6 nm in length will deform a total amount of 1.5 Å (under maximum physiologic load), which corresponds to a strain of around 0.59%. This amount of strain is extremely small to account for in stiffness calculations. Under torsional load though, actin appears to have a lot more flexibility, especially in regards to the change in φ_d. This data contradicts previously derived conclusions from both experiment and normal mode analyses that state F-actin is a relatively isotropic material in terms of stretching, bending and twisting [19, 77]. This raises the question of whether the appropriate residual metric for torsional loading was selected, were the optimum r_c and *k* parameters used, or simply does the SENM model do a poor job in representing the F-actin's deformation under torsional loads? In fact, the RMSD for mer5 between the two loading scenarios is 5 fold, which further shows that it is possible that the optimum r_c and *k* parameters were not selected due to the drastically decreased stiffness of the system.

From the SENM simulations, there exists a plethora of information for the elemental stresses and strains; however it is difficult to translate this information into usable data that could describe the protein's inter-residue interactions. One of the reasons the ANM is so simple is because it assigns one type of spring element connection between all residues that are within a predefined cutoff radius. It is this unbiased assignment that 'homogenizes' the protein, where some residues are connected to anywhere between 5 and 15 of their neighbors. However, realistically these connections carry no true meaning to the protein's embedded interactions. This is one research area

that needs further investigation in order to extract meaningful data from the element springs.

Another point to be made from the results of the axial and torsional loading scenarios on actin is that many of the regions on mer5 that undergo higher displacements also correspond to regions of increased protein flexibility. One of the original reasons ENMs were developed was because of the link between a protein's unloaded fluctuations to its functional conformational changes [39, 41, 78]. As was concluded by many of these researchers, protein regions with loop structures, which have fewer non-bonded interactions with their neighbors, tend to have a increased protein flexibility than with other more stable secondary structures (such as α-helices and β-sheets). One of the recurrent observations noticed in both the axial and torsional loading scenarios was the fact that protein regions characterized by loop conformations were experiencing higher displacements. Intuitively, this makes sense since a structure with a higher flexibility should exhibit a larger deformation to a load, just as a plastic rod will bend more than a metal rod. In that respect, the SENM does a great job in modeling those regions of protein structure.

To summarize, the previously presented data shows that the SENM can be used to determine sub-protein level details under certain loading conditions. The B1 and B2 QoIs show that the SENM cannot conclusively predict deformations between 2 non-covalently bonded residues that are in close proximity (<1.0 nm) to each other. However, under axial loading, the SENM model captures the intermer distance (>5.0 nm) and the dihedral angle well. Therefore, larger tertiary properties can be extracted using the SENM model. If residue level details about a loaded protein structure are desired, the

results of this thesis show that this task is best left to the higher resolution available from SMD simulations.

4.7 SUMMARY

To conclude this chapter, the results section has presented the unloaded 9mer model results, along with the parameterization of the structure under axial and torsional loading. The displacements from the SMD and SENM models were presented, in addition to the predefined QoIs that were quantified to see the SENM's effectiveness in predicting the 9mer structural response to load at the residue level. The chapter was concluded by a discussion of the results including a comparison between the SMD and SENM models.

Chapter 5

Conclusions and Future Work

5.1 Effectiveness of the SENM model

This thesis has presented an extensive amount of work in applying the SENM to a 9mer filament of actin under axial and torsional loading. And while the motivation for studying F-actin with this model has been clearly defined, the breadth of available proteins this model can be applied to extend far beyond actin and its related proteins. That being said, an obvious application of this model would be to study the well known actin binding motor protein myosin II, mainly because myosin's kinetics heavily depend on the applied strain to the motor domain region [79-81]. Clearly though, other structures such as microtubules and the anchoring proteins in focal adhesions all present themselves as proteins whose structural response to mechanical stimuli is highly desired and is the focus of so much cutting edge research.

In addition to the aforementioned applications of the SENM, the effectiveness of the model could be increased if loads that are applied to the protein system are large enough to cause deformations that are out of the RMSD range (>2.0 Å) of standard MD simulations alone. Loading a protein in this manner will ensure that the observed deformations that both the SMD and SENM simulations exhibit will be out of the range of the 'noise' that MD/SMD simulations are subject to. This process has to be executed with extreme care though because if the applied load is too large, a conformational change could be induced in the SMD system. If in fact it turns out that MD structures are

undergoing subtle conformational changes as opposed to simple linear rearrangements of the structure, then it is possible that multiple ENM's need to be adopted in order to account for different rearrangements for different load sizes. While for the scope of this project the SENM model has been able to capture a great deal of the desired QoIs, a larger level of deformation would give greater confidence to the overall robustness of the model, but at the same time cause an even greater mismatch due to an unforeseen conformational change.

As mentioned, there is a large discrepancy between the magnitude of displacements between the SMD simulations and the SENM model. However, in both the axial and torsional loading conditions, the SENM model is still able to capture regions that have relatively high and low displacements, such as the DB-loop, residues 240-248, 285-290,and 166-170. When it comes to quantifying more specific regions as outlined in the QoI table, it still reproduces the SMD deformations within the region of 1-σ.

5.2 FUTURE WORK

Working towards the goal of creating a structural network model that can capture both the mechanical deformations from an applied load and still emulate a protein's inter-residue connections, the next iteration of this work will be to break the inter-residue interactions into two categories: bonded (covalent) and non-bonded (hydrogen bonding, electrostatic interactions, etc…). This will combat the issue observed with the intermer distance QoI because of the 'weaker region' that develops between monomers during element assignments. Doing so will allow for a more realistic potential between the non-covalently bonded monomers.

These network properties are best captured by the chemical network model, previously mentioned in Table 1 of section 2.4.3, as was pioneered by Kondrashov, et al. [61]. While the model is slightly more complex than the GNM/ANM, it will allow for (a): a more realistic potential that exists between 2 non-bonded monomers, and (b) a more quantitative assessment of inter-residue stresses and strains because of the fact that residues will now be connected in a manner that physically mimics true protein chemistry.

Looking beyond the improvements to the SENM, the SMD simulations would benefit from further refinement and iterations as well. In other G and F-actin studies involving MD simulations, it is common for the simulation time to run anywhere from 50 ns up to 100 ns [22, 23]. While these simulations were performed on an unloaded structure, it is reasonable to assume that the addition of external loads would necessitate additional simulation time in order to allow for the load to propagate through the entire protein structure. It is a very real possibility that the final structures used for the SMD results and SENM parameterization were taken from a structure that was not fully extended or twisted (depending on the loading scenario). A simulation time of 10 ns, while appropriately set by the lab's available computational resources, seems too short to allow for a structure of this size to fully equilibrate after the loads were applied.

In summation, the study of the structural response of proteins to external loads using an elastic network model is still a relatively new and exciting topic in the field of structural biology. As more and more information regarding protein signaling pathways continues to surface, the desire to learn and understand more about these underlying

processes will only increase as the thirst for knowledge and truth are quenched by their discoveries.

Appendix A

FORTRAN 90 Normal Mode Analysis Code Modifications

test_run_eigens.f90

```
PROGRAM main

IMPLICIT NONE

REAL(KIND=8) :: rc, gamma, misfit

!_____________________________________________________

rc = 19.D0
gamma = 9.49D0

CALL eigen_solver(rc,gamma,misfit)

write(*,*) 'misfit=', misfit

END PROGRAM main
```

```
!PROGRAM eigen_solver
SUBROUTINE eigen_solver(rc,gamma,misfit)

!USE netcdf
USE util

IMPLICIT NONE

INTEGER, PARAMETER :: INP=10, LU=11, OUT=12, INP2=13
INTEGER :: i, j, N, cntr, ierror, info
INTEGER :: nEigVs, ldz
INTEGER :: case

REAL(KIND=8) :: misfit , rc, gamma, k

CHARACTER(len=1) :: eigVec, allEigVals
CHARACTER(len=30) :: inputFile, outputFile, inputFile_delta
CHARACTER(len=30) :: filename1, filename2
!-------------------------------------------------------
!--------------------- NetCDF stuff -------------------
INTEGER :: ncid
INTEGER :: evs_dimid, evs_varid
INTEGER :: evs_def_dimid, evs_def_varid
INTEGER :: eigval_dimid, eigvec_dimid, eigvec_varid
INTEGER :: eigval_def_dimid, eigvec_def_dimid, eigvec_def_varid
INTEGER, DIMENSION(2) :: dimids
!_____________________________________________________________
!!$OPEN(UNIT=INP2, FILE='temp_out.txt', STATUS='old', ACTION='read',
IOSTAT=ierror)
!!$read(INP2,*) rc, gamma
!!$CLOSE(INP2)

OPEN(UNIT=INP, FILE='inputfile.txt', STATUS='old', ACTION='read',
IOSTAT=ierror)

DO i = 1, 32
   IF (i == 2) THEN
      READ(INP,*,IOSTAT=ierror) inputFile
!      write(*,*) inputFile
   ELSEIF (i == 5)  THEN
      READ(INP,*,IOSTAT=ierror) outputFile
!      write(*,*) outputFile
   ELSEIF (i == 8)  THEN
      READ(INP,*,IOSTAT=ierror) inputFile_delta
!      write(*,*) inputFile_delta
   ELSEIF (i == 11)  THEN
      READ(INP,*,IOSTAT=ierror) !inputFile_ext_loads
!      write(*,*) inputFile_ext_loads
   ELSEIF (i == 14) THEN
      READ(INP,*,IOSTAT=ierror) eigVec
!      write(*,*) eigVec
   ELSEIF (i == 17) THEN
```

```
      READ(INP,*,IOSTAT=ierror) nEigVs
!      write(*,*) nEigVs
   ELSEIF (i == 20) THEN
      READ(INP,*,IOSTAT=ierror) !rc              !will be an input from
QUESO
!      write(*,*) 'rc=', rc
   ELSEIF (i == 23) THEN
      READ(INP,*,IOSTAT=ierror) !gamma           !will be an input from
QUESO
!      write(*,*) 'k=', gamma
   ELSEIF (i == 26) THEN
      READ(INP,*,IOSTAT=ierror) filename2
!      write(*,*) 'filename2: ', filename2
   ELSEIF (i == 29) THEN
      READ(INP,*,IOSTAT=ierror) T
!      write(*,*) 'T= ', T
   ELSEIF (i == 32) THEN
      READ(INP,*,IOSTAT=ierror) case
!      write(*,*) 'case= ', case
   ELSE
      READ(INP,*,IOSTAT=ierror)
   END IF

   IF (ierror /= 0) THEN
      write(*,*) 'ierror=', ierror, 'i=', i
      STOP
   END IF
END DO

CLOSE(INP)

!-----------------------------------------------------
!---- Calculate the eigenvalues and eignevectors ----
!---- for the "UNDEFORMED" structure             ----
!-----------------------------------------------------
CALL calc_hess(inputFile,rc,gamma)

N = INT(hessian(SIZE(hessian,1),1))
ldummy=.TRUE.

CALL get_eigens(N,rc,gamma)
CALL calc_B_factors(N,gamma)
CALL calc_residual(filename2,B_factors,misfit,N,gamma,case,k,w,z)

IF (ALLOCATED(w)) DEALLOCATE(w)
IF (ALLOCATED(z)) DEALLOCATE(z)
IF (ALLOCATED(posCA)) DEALLOCATE(posCA)
IF (ALLOCATED(hessian)) DEALLOCATE(hessian)
IF (ALLOCATED(B_factors)) DEALLOCATE(B_factors)

write(*,"('',F10.4)") misfit

END SUBROUTINE eigen_solver
!END PROGRAM eigen_solver
```

```
!---------------------------------------------------------------------
!---- Write the undeformed structure's eigenvalies and eigenvectors ---
!---------------------------------------------------------------------
!!$ldz = N
!!$
!!$CALL check( nf90_create(outputfile, NF90_CLOBBER, ncid) )
!!$
!!$CALL check( nf90_def_dim(ncid, "Eigenvalue_Number", N, eigval_dimid) )
!!$CALL check( nf90_def_dim(ncid, "Eigenvectors_Components", ldz,
eigvec_dimid) )
!!$
!!$CALL check( nf90_def_dim(ncid, "Eigenvalues", N, evs_dimid) )
!!$
!!$dimids =  (/  eigval_dimid , eigvec_dimid  /)
!!$
!!$CALL check( nf90_def_var(ncid, "Eigenvectors", NF90_DOUBLE, dimids,
eigvec_varid) )
!!$
!!$CALL check( nf90_def_var(ncid, "Eigenvalues", NF90_DOUBLE, evs_dimid,
evs_varid) )
!!$
!!$CALL check( nf90_def_dim(ncid, "Deformed_Eigenvalue_Number", N,
eigval_def_dimid) )
!!$
!!$CALL check( nf90_def_dim(ncid, "Deformed_Eigenvectors_Components", ldz,
eigvec_def_dimid) )
!!$
!!$CALL check( nf90_def_dim(ncid, "Deformed_Eigenvalues", N, evs_def_dimid)
)
!!$
!!$dimids =  (/  eigval_def_dimid , eigvec_def_dimid  /)
!!$
!!$CALL check( nf90_def_var(ncid, "Deformed_Eigenvectors", NF90_DOUBLE,
dimids, eigvec_def_varid) )
!!$
!!$CALL check( nf90_def_var(ncid, "Deformed_Eigenvalues", NF90_DOUBLE,
evs_def_dimid, evs_def_varid) )
!!$
!!$CALL check( nf90_enddef(ncid) )
!!$
!!$CALL check( nf90_put_var(ncid, eigvec_varid, TRANSPOSE(z)))
!!$
!!$CALL check( nf90_put_var(ncid, evs_varid, w))
!!$!-------------------------------------------------
!!$!---- Calculate the eigenvalues and eignevectors ----
!!$!---- for the "DEFORMED" structure               ----
!!$!-------------------------------------------------
!!$CALL calc_hess_deformed(inputFile_delta)
!!$
!!$CALL check( nf90_put_var(ncid, eigvec_varid, TRANSPOSE(z)))
!!$
!!$CALL check( nf90_put_var(ncid, evs_varid, w))
!!$
!!$CALL get_eigens(N)
```

```
!!$
!!$!-----------------------------------------------
!!$CALL check( nf90_put_var(ncid, eigvec_def_varid, TRANSPOSE(z)))
!!$
!!$CALL check( nf90_put_var(ncid, evs_def_varid, w))
!!$
!!$CALL check( nf90_close(ncid) )
!!$
!!$IF (ALLOCATED(posCA)) DEALLOCATE(posCA)
!!$
!!$!-----------------------------------------------
!!$!---- Writing the eigenvalues and eigenvectors ----
!!$!---- into the output file using NETCDF        ----
!!$!-----------------------------------------------
!!$!CALL write_to_file(N,ldz,w,z,w,z,outputFile)
!!$!CALL write_to_file(N,ldz,w_undef,z_undef,w,z,outputFile)
!!$write(*,*) '**** Eigenvalues and eigenvectors are calculated'
!!$write(*,*) '     and written to the output file: ', outputFile
!!$
!!$!_____________________________________________________________________________
!!$!_____________________________________________________________________________
!!$
!!$CONTAINS
!!$  SUBROUTINE CHECK(status)
!!$    INTEGER, INTENT(IN) :: status
!!$
!!$    IF(status /= nf90_noerr) THEN
!!$      print *, trim(nf90_strerror(status))
!!$      STOP "Stopped"
!!$    END IF
!!$  END SUBROUTINE CHECK
!!$END PROGRAM eigen_solver


!ifort -I$TACC_MKL_INC -L$TACC_MKL_LIB -lmkl_em64t -lmkl -lguide -lpthread
-lmkl_lapack util.f90 get_eigens.f90 write_to_file.f90 eigen_solver.f90 -
traceback -CB -o ./a2.out -L$TACC_NETCDF_LIB -I$TACC_NETCDF_INC -lnetcdf
```

Util.f90

```
MODULE util

IMPLICIT NONE

SAVE

INTEGER :: numCA

!REAL(KIND=8) :: rc=15      !cutoff radius
!REAL(KIND=8) :: gamma=20   !spring constant
REAL(KIND=8) :: T       !Temperature
REAL(KIND=8), ALLOCATABLE, DIMENSION(:)   :: w
REAL(KIND=8), ALLOCATABLE, DIMENSION(:,:) :: z
REAL(KIND=8), ALLOCATABLE, DIMENSION(:,:) :: posCA
REAL(KIND=8), ALLOCATABLE, DIMENSION(:,:) :: hessian
REAL(KIND=8), ALLOCATABLE, DIMENSION(:)   :: B_factors
REAL(KIND=8), ALLOCATABLE, DIMENSION(:)   :: w_undef
REAL(KIND=8), ALLOCATABLE, DIMENSION(:,:) :: z_undef

CHARACTER(len=30) :: inputFile_ext_loads

LOGICAL :: ldummy=.TRUE.
!_____________________________________________________________________
!/////////////////////////////////////////////////////////////////////
!_____________________________________________________________________

CONTAINS

SUBROUTINE calc_hess(fileName,rc,gamma)

IMPLICIT NONE

!-----------------------
!---- INPUTS/OUTPUT ----
!-----------------------
REAL(KIND=8) :: rc, gamma
CHARACTER(len=30), INTENT(IN) :: fileName

!-------------------------
!---- Local Variables ----
!-------------------------
INTEGER, parameter :: LU = 10
INTEGER :: idummy1, idummy2
INTEGER :: i, j, ierror, cntr, cntr2

REAL(KIND=8) :: ddummy1, ddummy2, ddummy3
REAL(KIND=8) :: dx, dy, dz, distsqr
REAL(KIND=8), ALLOCATABLE, DIMENSION(:,:) :: delta_posCA

CHARACTER(len=30) :: cdummy1, cdummy2, cdummy3, cdummy4
!_____________________________________________________________________
```

```
!IF (ldummy) THEN
   numCA = 0

   OPEN(UNIT=LU, FILE=fileName, STATUS='old', ACTION='read', IOSTAT=ierror)

   DO
      READ(LU,*,IOSTAT=ierror)
      IF (ierror /= 0) EXIT
      numCA = numCA + 1
   END DO

   rewind(LU)

   ALLOCATE(posCA(3,numCA))
   posCA = 0.D0

   DO i = 1, numCA
      READ(LU,*,IOSTAT=ierror) ddummy1, ddummy2, ddummy3
      posCA(1,i) = ddummy1
      posCA(2,i) = ddummy2
      posCA(3,i) = ddummy3
   END DO

   close(LU)

   ldummy = .FALSE.

!   write(*,*) '1. finished reading the PDB file'
!ELSE

!   ALLOCATE(delta_posCA(SIZE(posCA,1),SIZE(posCA,2)))
!   delta_posCA = 0.D0

!   OPEN(UNIT=LU, FILE=fileName, STATUS='old', ACTION='read',
IOSTAT=ierror)

!   DO i = 1, numCA
!      READ(LU,*,IOSTAT=ierror) delta_posCA(1,i), delta_posCA(2,i),
delta_posCA(3,i)
!   END DO

!   close(LU)

!   posCA = posCA + delta_posCA

!   IF (ALLOCATED(delta_posCA)) DEALLOCATE(delta_posCA)

!   write(*,*) '4. finished reading the displacement input file'
!END IF

!----------------------------------------
!---- Calculating the Hessian matrix ----
!----------------------------------------
cntr = 0
```

```
DO j = 1, numCA
   DO i = j+1, numCA

      dx = posCA(1,i) - posCA(1,j)
      dy = posCA(2,i) - posCA(2,j)
      dz = posCA(3,i) - posCA(3,j)
      distsqr = dx*dx + dy*dy + dz*dz

      IF (distsqr < rc*rc) cntr = cntr + 9
   END DO
END DO

cntr2 = cntr
cntr = numCA*6 + cntr

ALLOCATE(hessian(cntr,3))
hessian = 0.D0

cntr = 0

DO i = 1, numCA
   DO j = 1, numCA
      IF (i == j) CYCLE

      dx = posCA(1,i) - posCA(1,j)
      dy = posCA(2,i) - posCA(2,j)
      dz = posCA(3,i) - posCA(3,j)
      distsqr = dx*dx + dy*dy + dz*dz

      IF (distsqr < rc*rc) THEN
         IF (j>i) THEN
            cntr = cntr + 9

            !Off-Diagonal Super-Elements
            hessian(cntr-8,1) = 3*i-2
            hessian(cntr-7,1) = 3*i-1
            hessian(cntr-6,1) = 3*i
            hessian(cntr-5,1) = 3*i-2
            hessian(cntr-4,1) = 3*i-2
            hessian(cntr-3,1) = 3*i-1
            hessian(cntr-2,1) = 3*i-1
            hessian(cntr-1,1) = 3*i
            hessian(cntr  ,1) = 3*i

            hessian(cntr-8,2) = 3*j-2
            hessian(cntr-7,2) = 3*j-1
            hessian(cntr-6,2) = 3*j
            hessian(cntr-5,2) = 3*j-1
            hessian(cntr-4,2) = 3*j
            hessian(cntr-3,2) = 3*j-2
            hessian(cntr-2,2) = 3*j
            hessian(cntr-1,2) = 3*j-2
            hessian(cntr  ,2) = 3*j-1
```

```
            hessian(cntr-8,3) = -dx*dx/distsqr
            hessian(cntr-7,3) = -dy*dy/distsqr
            hessian(cntr-6,3) = -dz*dz/distsqr
            hessian(cntr-5,3) = -dx*dy/distsqr
            hessian(cntr-4,3) = -dx*dz/distsqr
            hessian(cntr-3,3) = -dy*dx/distsqr
            hessian(cntr-2,3) = -dy*dz/distsqr
            hessian(cntr-1,3) = -dz*dx/distsqr
            hessian(cntr  ,3) = -dz*dy/distsqr
         END IF

         !Diagonal Super-Elements
         hessian(cntr2+6*j-5,3) = hessian(cntr2+6*j-5,3) + dx*dx/distsqr
         hessian(cntr2+6*j-4,3) = hessian(cntr2+6*j-4,3) + dx*dy/distsqr
         hessian(cntr2+6*j-3,3) = hessian(cntr2+6*j-3,3) + dx*dz/distsqr
         hessian(cntr2+6*j-2,3) = hessian(cntr2+6*j-2,3) + dy*dy/distsqr
         hessian(cntr2+6*j-1,3) = hessian(cntr2+6*j-1,3) + dy*dz/distsqr
         hessian(cntr2+6*j  ,3) = hessian(cntr2+6*j  ,3) + dz*dz/distsqr

      END IF
   END DO
END DO

DO i = 1, numCA
   cntr = cntr + 6

   hessian(cntr-5,1) = 3*i-2
   hessian(cntr-4,1) = 3*i-2
   hessian(cntr-3,1) = 3*i-2
   hessian(cntr-2,1) = 3*i-1
   hessian(cntr-1,1) = 3*i-1
   hessian(cntr  ,1) = 3*i

   hessian(cntr-5,2) = 3*i-2
   hessian(cntr-4,2) = 3*i-1
   hessian(cntr-3,2) = 3*i
   hessian(cntr-2,2) = 3*i-1
   hessian(cntr-1,2) = 3*i
   hessian(cntr  ,2) = 3*i

END DO

!write(*,*) '2. finished creating the Hessian Matrix for the UNDEFORMED
structure'

END SUBROUTINE calc_hess
!_____________________________________________________________________
!_____________________________________________________________________
!_____________________________________________________________________
SUBROUTINE calc_hess_deformed(fileName,rc,gamma)

IMPLICIT NONE
```

```
!-----------------------
!---- INPUTS/OUTPUT ----
!-----------------------
REAL(KIND=8) :: rc, gamma
CHARACTER(len=30), INTENT(IN) :: fileName

!-------------------------
!---- Local Variables ----
!-------------------------
INTEGER, parameter :: LU = 10
INTEGER :: idummy1, idummy2
INTEGER :: i, j, ierror, cntr, cntr2

REAL(KIND=8) :: ddummy1, ddummy2, ddummy3
REAL(KIND=8) :: dx, dy, dz, dist, distsqr
REAL(KIND=8) :: dx0, dy0, dz0, dist0
REAL(KIND=8) :: dist_ratio
REAL(KIND=8), ALLOCATABLE, DIMENSION(:,:) :: delta_posCA

CHARACTER(len=30) :: cdummy1, cdummy2, cdummy3, cdummy4
!_____________________________________________________________________


ALLOCATE(delta_posCA(SIZE(posCA,1),SIZE(posCA,2)))
delta_posCA = 0.D0

OPEN(UNIT=LU, FILE=fileName, STATUS='old', ACTION='read', IOSTAT=ierror)
IF (ierror /= 0) THEN
   write(*,*) 'problem opening the file', fileName
STOP "Stopping"
END IF

DO i = 1, numCA
   READ(LU,*,IOSTAT=ierror) idummy1, delta_posCA(1,i), delta_posCA(2,i),
delta_posCA(3,i)
END DO

close(LU)

delta_posCA = 0.D0
delta_posCA = posCA + delta_posCA

!write(*,*) '4. finished reading the displacement input file'

!---------------------------------------
!---- Calculating the Hessian matrix ----
!---------------------------------------
cntr = 0

DO j = 1, numCA
   DO i = j+1, numCA

      dx = posCA(1,i) - posCA(1,j)
      dy = posCA(2,i) - posCA(2,j)
```

```
      dz = posCA(3,i) - posCA(3,j)
      distsqr = dx*dx + dy*dy + dz*dz

      IF (distsqr < rc*rc) cntr = cntr + 9
   END DO
END DO

cntr2 = cntr
cntr = numCA*6 + cntr

ALLOCATE(hessian(cntr,3))
hessian = 0.D0

cntr = 0

DO i = 1, numCA
   DO j = 1, numCA
      IF (i == j) CYCLE

      dx = delta_posCA(1,i) - delta_posCA(1,j)
      dy = delta_posCA(2,i) - delta_posCA(2,j)
      dz = delta_posCA(3,i) - delta_posCA(3,j)
      dist = SQRT(dx*dx + dy*dy + dz*dz)

      IF (dist < rc) THEN

         dx0 = posCA(1,i) - posCA(1,j)
         dy0 = posCA(2,i) - posCA(2,j)
         dz0 = posCA(3,i) - posCA(3,j)
         dist0 = SQRT(dx0*dx0 + dy0*dy0 + dz0*dz0)

         dist_ratio = dist0/(dist*dist*dist)

         IF (j>i) THEN
            cntr = cntr + 9

            !Off-Diagonal Super-Elements
            hessian(cntr-8,1) = 3*i-2
            hessian(cntr-7,1) = 3*i-1
            hessian(cntr-6,1) = 3*i
            hessian(cntr-5,1) = 3*i-2
            hessian(cntr-4,1) = 3*i-2
            hessian(cntr-3,1) = 3*i-1
            hessian(cntr-2,1) = 3*i-1
            hessian(cntr-1,1) = 3*i
            hessian(cntr  ,1) = 3*i

            hessian(cntr-8,2) = 3*j-2
            hessian(cntr-7,2) = 3*j-1
            hessian(cntr-6,2) = 3*j
            hessian(cntr-5,2) = 3*j-1
            hessian(cntr-4,2) = 3*j
            hessian(cntr-3,2) = 3*j-2
            hessian(cntr-2,2) = 3*j
```

```
            hessian(cntr-1,2) = 3*j-2
            hessian(cntr  ,2) = 3*j-1

            hessian(cntr-8,3) = -1.D0 + dist0/dist - dx*dx*dist_ratio
!-dx*dx/distsqr
            hessian(cntr-7,3) = -1.D0 + dist0/dist - dy*dy*dist_ratio
!-dy*dy/distsqr
            hessian(cntr-6,3) = -1.D0 + dist0/dist - dz*dz*dist_ratio
!-dz*dz/distsqr
            hessian(cntr-5,3) = -dx*dy*dist_ratio     !-dx*dy/distsqr
            hessian(cntr-4,3) = -dx*dz*dist_ratio     !-dx*dz/distsqr
            hessian(cntr-3,3) = -dy*dx*dist_ratio     !-dy*dx/distsqr
            hessian(cntr-2,3) = -dy*dz*dist_ratio     !-dy*dz/distsqr
            hessian(cntr-1,3) = -dz*dx*dist_ratio     !-dz*dx/distsqr
            hessian(cntr  ,3) = -dz*dy*dist_ratio     !-dz*dy/distsqr
         END IF

         !Diagonal Super-Elements
         hessian(cntr2+6*j-5,3) = hessian(cntr2+6*j-5,3) + 1.D0 -
dist0/dist + dx*dx*dist_ratio     !dx*dx/distsqr
         hessian(cntr2+6*j-4,3) = hessian(cntr2+6*j-4,3) + dx*dy*dist_ratio
!dx*dy/distsqr
         hessian(cntr2+6*j-3,3) = hessian(cntr2+6*j-3,3) + dx*dz*dist_ratio
!dx*dz/distsqr
         hessian(cntr2+6*j-2,3) = hessian(cntr2+6*j-2,3) + 1.D0 -
dist0/dist + dy*dy*dist_ratio     !dy*dy/distsqr
         hessian(cntr2+6*j-1,3) = hessian(cntr2+6*j-1,3) + dy*dz*dist_ratio
!dy*dz/distsqr
         hessian(cntr2+6*j  ,3) = hessian(cntr2+6*j  ,3) + 1.D0 -
dist0/dist + dz*dz*dist_ratio     !dz*dz/distsqr

      END IF
   END DO
END DO

DO i = 1, numCA
   cntr = cntr + 6

   hessian(cntr-5,1) = 3*i-2
   hessian(cntr-4,1) = 3*i-2
   hessian(cntr-3,1) = 3*i-2
   hessian(cntr-2,1) = 3*i-1
   hessian(cntr-1,1) = 3*i-1
   hessian(cntr  ,1) = 3*i

   hessian(cntr-5,2) = 3*i-2
   hessian(cntr-4,2) = 3*i-1
   hessian(cntr-3,2) = 3*i
   hessian(cntr-2,2) = 3*i-1
   hessian(cntr-1,2) = 3*i
   hessian(cntr  ,2) = 3*i

END DO
```

```
!IF (ALLOCATED(delta_posCA)) DEALLOCATE(delta_posCA)

write(*,*) '5 finished creating the Hessian Matrix for the DEFORMED
structure'

END SUBROUTINE calc_hess_deformed
!_____________________________________________________________________
!_____________________________________________________________________
!_____________________________________________________________________

SUBROUTINE calc_B_factors(N,gamma)

INTEGER :: N
INTEGER :: i, j

REAL(KIND=8) :: k
REAL(KIND=8) :: tr
REAL(KIND=8) :: gamma
REAL(KIND=8), PARAMETER :: pi=3.14159265, kB=1.38D-23
!REAL(KIND=8), ALLOCATABLE, DIMENSION(:) :: B_factors
REAL(KIND=8), ALLOCATABLE, DIMENSION(:) :: inv_hess
!_____________________________________________________
ALLOCATE(inv_hess(N))
inv_hess = 0.D0

DO i = 7, N
   DO j = 1, N
      inv_hess(j) = inv_hess(j) + (1/w(i))*z(j,i)*z(j,i)
   END DO
END DO

ALLOCATE(B_factors(N/3))
B_factors = 0.D0

tr = 0.D0
DO i = 1, N/3
   tr = inv_hess(3*i-2) + inv_hess(3*i-1) + inv_hess(3*i)
   B_factors(i) = 8.D0*pi*pi*tr/(3.D0*gamma*1.D-2)
END DO

IF (ALLOCATED(inv_hess)) DEALLOCATE(inv_hess)

END SUBROUTINE calc_B_factors
!_____________________________________________________________________

END MODULE util

!cat 1atp.pdb | grep -w ATOM | grep CA > filtered_pdb_1atp.txt
```

```
SUBROUTINE get_eigens(N,rc,gamma)

USE util
!USE omp_lib

IMPLICIT NONE

INTEGER :: nt, lda
INTEGER :: i, j, k, N, info
INTEGER :: ldz, lwork, liwork
INTEGER, ALLOCATABLE, DIMENSION(:) :: iwork

REAL(KIND=8) :: rc, gamma
REAL(KIND=8) :: ti,tf, gtod_timer
REAL(KIND=8), ALLOCATABLE, DIMENSION(:)   :: work
REAL(KIND=8), ALLOCATABLE, DIMENSION(:)   :: ap
REAL(KIND=8), ALLOCATABLE, DIMENSION(:,:) :: a

CHARACTER(len=1) :: uplo, jobz
!_____________________________________________________________
!ti = gtod_timer();

!! nt = 1
!! write(*,*) 'Number of Threads (before) = ', nt
!! !$omp parallel
!! !$   nt = omp_get_num_threads()
!! !$omp master
!!    write(*,*) 'Number of Threads (after, in the loop) = ', nt
!! !$omp end master
!! !$omp end parallel

!! write(*,*) 'Number of Threads (after) = ', nt

!-----------------------------------------------
!---- Making the array AP that contains the ----
!---- section of matrix A lower triangular. ----
!----       AP is in Packed Format          ----
!-----------------------------------------------

!ALLOCATE(ap(N*(N+1)/2))
!ap = 0.D0
ALLOCATE(a(N,N))
a = 0.D0

!$omp parallel do PRIVATE(i,j)
DO k = 1, SIZE(hessian,1)
   j = INT(hessian(k,1))
   i = INT(hessian(k,2))
   a(j,i) = hessian(k,3)
!   ap(i+(2*N-j)*(j-1)/2) = hessian(k,3)
END DO
!$omp end parallel do
```

```
IF (ALLOCATED(hessian)) DEALLOCATE(hessian)

!---------------------------------------------------------------------
!---- Adding the external spring constants to the Hessian Matrix ----
!---------------------------------------------------------------------
!IF (ldummy) THEN
!   ldummy = .FALSE.
!ELSE
!   k = SIZE(ap)
!   CALL add_load_to_hessian(ap,k,inputFile_ext_loads,N,gamma)
!   CALL add_load_to_hessian(a,N,inputFile_ext_loads,N,gamma)
!END IF

!-------------------------------------------------------------
!---- Calculating the eigenvalues of the Hessian Matrix ----
!-------------------------------------------------------------

IF (ALLOCATED(w)) DEALLOCATE(w)
IF (ALLOCATED(z)) DEALLOCATE(z)

uplo = 'U'
jobz = 'V'

!ldz = N
!ALLOCATE(z(ldz,N))
!z = 0.D0

lda = N
ALLOCATE(z(lda,N))
z = 0.D0

ALLOCATE(w(N))
w = 0.D0

lwork = 2*N*N + 6*N + 1
ALLOCATE(work(lwork))
work = 0.D0

liwork = 5*N + 3
ALLOCATE(iwork(liwork))
iwork = 0

call dsyevd(jobz, uplo, n, a, lda, w, work, lwork, iwork, liwork, info)

!call dspevd(jobz, uplo, n, ap, w, z, ldz, work, lwork, iwork, liwork,
info)
!tf = gtod_timer();

!write(*,*) "total time elapsed= ", real(tf-ti)
!write(*,*) '3. finished calculating the eigs'

z = a
```

```
IF (ALLOCATED(iwork)) DEALLOCATE(iwork)
IF (ALLOCATED(work)) DEALLOCATE(work)
!IF (ALLOCATED(ap)) DEALLOCATE(ap)
IF (ALLOCATED(a)) DEALLOCATE(a)

END SUBROUTINE get_eigens
```

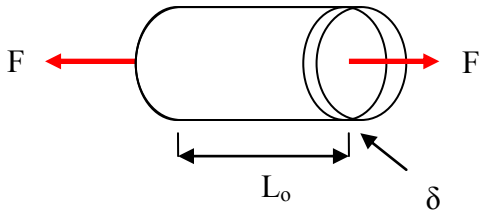
```
!SUBROUTINE calc_residual(GS_filename,B_factors,misfit,N)
SUBROUTINE calc_residual(GS_filename,data,misfit,N,gamma,case,k,w,z)

IMPLICIT NONE

INTEGER, PARAMETER :: UPDBB=20
INTEGER :: i, j, ierror
INTEGER :: N, m
INTEGER :: case

REAL(KIND=8), PARAMETER :: pi=3.14159265, kbt_correct=41.1D0
REAL(KIND=8) :: mean_data, mean_GS
REAL(KIND=8) :: std_data, std_GS
REAL(KIND=8) :: e, misfit, tr, bf_sum
REAL(KIND=8) :: k, gamma
REAL(KIND=8), DIMENSION(N/3) :: GS_data       !PDB_B_factors
REAL(KIND=8), DIMENSION(N/3) :: data          !B_factors
REAL(KIND=8), DIMENSION(N)   :: w
REAL(KIND=8), DIMENSION(N,N) :: z
REAL(KIND=8), ALLOCATABLE, DIMENSION(:) :: inv_hess

CHARACTER(len=30) :: GS_filename
!_____________________________________________________

OPEN(UNIT=UPDBB,FILE=GS_filename,STATUS='old',ACTION='read',IOSTAT=ierror)
IF (ierror /= 0) THEN
   write(*,*) 'A problem occured when trying to open the file ',
GS_filename
   write(*,*) 'IOSTAT=', ierror
   STOP
END IF

m = N/3

GS_data = 0.D0
DO i = 1, m
   READ(UPDBB,*) GS_data (i) !PDB_B_factors(i)
END DO

CLOSE(UNIT=UPDBB)

misfit = 0.D0

IF (case == 1) THEN
   DO i = 1, m
      misfit = misfit + (data(i) - GS_data(i))*(data(i) - GS_data(i))
   END DO

ELSE IF (case == 2) THEN
   mean_data = SUM(data)/m
   mean_GS   = SUM(GS_data)/m
```

```
   std_data = 0.D0
   std_GS   = 0.D0
   DO i = 1, m
      std_data = std_data + (data(i) - mean_data)*(data(i) - mean_data)
      std_GS   = std_GS  + (GS_data(i) - mean_GS)*(GS_data(i) - mean_GS)
   END DO

   std_data = SQRT(std_data/m)
   std_GS   = SQRT(std_GS/m)

   DO i = 1, m
      misfit = misfit + (data(i) - mean_data)*(GS_data(i) - mean_GS)
   END DO

   misfit = 1.D0 - misfit/(std_data*std_GS*(m-1))

ELSE IF (case == 3) THEN

   !------------------------------
   !---- Calculate Correlation ----
   !------------------------------
   mean_data = SUM(data)/m
   mean_GS   = SUM(GS_data)/m

   std_data = 0.D0
   std_GS   = 0.D0
   DO i = 1, m
      std_data = std_data + (data(i) - mean_data)*(data(i) - mean_data)
      std_GS   = std_GS  + (GS_data(i) - mean_GS)*(GS_data(i) - mean_GS)
   END DO

   std_data = SQRT(std_data/m)
   std_GS   = SQRT(std_GS/m)

   DO i = 1, m
      misfit = misfit + (data(i) - mean_data)*(GS_data(i) - mean_GS)
   END DO

   misfit = 1.D0 - misfit/(std_data*std_GS*(m-1))

   !-----------------------------
   !---- Calculate optimum k ----
   !-----------------------------
   ALLOCATE(inv_hess(N))
   inv_hess = 0.D0

   DO i = 7, N
      DO j = 1, N
         inv_hess(j) = inv_hess(j) + (1/w(i))*z(j,i)*z(j,i)
      END DO
   END DO

   tr = 0.D0
```

```
   DO i = 1, m
      tr = tr + inv_hess(3*i-2) + inv_hess(3*i-1) + inv_hess(3*i)
   END DO

   bf_sum = SUM(GS_data)

   k = 8.D0*pi*pi*tr*kbt_correct/(3.D0*bf_sum)

   misfit = misfit + ABS(k - gamma)/k

   IF (ALLOCATED(inv_hess)) DEALLOCATE(inv_hess)

ELSE
   write(*,*) 'The entry for "Comparison Criteria" in the "inputfile.txt"
is incorrect.'
   write(*,*) 'Please enter either "1" for RMSD, "2" for CORRELATION or'
   write(*,*) '"3" for BOTH  in the "inputfile.txt".'
   write(*,*) 'misfit cannot be calculated.'
   write(*,*) 'Program stops now.'
   STOP
END IF

misfit = -0.5D0*SQRT(misfit)

END SUBROUTINE calc_residual
```

Appendix B

Calculation for F-actin Extension under Axial Load



Governing Equations:

$$k = \frac{F}{\delta} = \frac{AE}{L_0} \tag{A.1}$$

We also know that the term *AE*, the cross sectional area *A* times Young's Modulus, *E,* is an extensive property of a material, therefore

$$AE = k_1 L_1 = k_2 L_2 \tag{A.2}$$

Where *AE* represents the true value presented from Liu et al.., since the given stiffness, or more appropriately, *AE*, for an F-actin length of 1 μm is 34.5 pN-μm/nm. So, after multiplying both sides of equation 1.1 by L_o and solving for the displacement, δ, we then have

$$\delta = \frac{FL_0}{AE} \tag{A.3}$$

Given values:

AE = 34.5 pN-μm/nm, from [17].

L_o = 25.4 nm (.0254 μm)

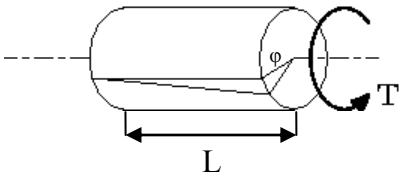
F = 200 pN

Plugging these values in to equation 1.3 yields

$$\delta = \frac{(200\ pN) * (.0254\ \mu m)}{34.5\ pN - \mu m/nm} = 0.147\ nm = 1.47\ \text{Å}$$

Appendix C

Calculation for F-actin Twist under Torsional Load



Governing Equation:

$$\varphi = \frac{TL}{GI_p} \tag{B.1}$$

Where GI_p = torsional rigidity

Given values:

$GI_p = 8.0 \pm 1.2 \text{ X } 10^{-26}$ Nm^2, from [19]

T = 200 pN-nm

L = 25.4 nm (254.0 Å)

Using dimensional analysis, GI_p = 80,000 pN-nm^2. Therefore

$$\varphi = \frac{(200\ pN - nm) * (25.4\ nm)}{80{,}000\ pN - nm^2} = 0.0636\ rad$$

From the characteristics of a circle, we know that displacement, u,

$$u = \varphi * r \tag{B.2}$$

where r, the radius of F-actin if approximated as a cylinder, is taken as 3.5 nm.

Finally, we have

$$u = (0.0636\, rad) * (3.5\, nm) = \ 0.223\, nm = 2.23\ \text{Å}$$

Appendix D

FORTRAN 90 Brute Force Residual Calculation Code

```
Program Main_rk
Use Abaqus_call, Only: &
     femrun, r_def, check_range, profit_run, rmsd_pdb, res_cal, Dali_Score
Implicit None

! ************************************************************** !
!                 Copywirght (c) IMPACT LAB, UT Austin           !
! -------------------------------------------------------------- !
! Chia-Cheng "Dennis" Liu                                        !
!----------------------------------------------------------------!
! Update      |     Modification                                 !
! 8/12/2010         start the system call ccode for Calculix     !
! 9/17/2010         Modify to include Dali similary scores       !
! ************************************************************** !

REAL*8 :: k
REAL*8 :: r_c
Real*8, Allocatable, Dimension(:,:) :: Res_set
Real*8 :: GS_Res(10), Ori_Res(9)

Integer :: N, N1, N_monomer
Integer ::  I, I_load,  Isa0, i_debug, irange(2), i_set,  I_prd, i_fit
LOGICAL :: file_exists

! spring constant of F-actin 34.5+-3.5
Real*8, parameter :: actin_k = 31.0d0

! filename system
CHARACTER (LEN=1) :: C2_INDEX
!CHARACTER (LEN=20) :: FILE_NAME
CHARACTER (LEN=10) :: BEN1, BEN2,ZONE_T
DOUBLE PRECISION :: K_INDEX
INTEGER :: TIME(8), KI_INDEX, T_INDEX
Character (Len=50) :: csv_file, filename, GS_file, fit_file, old_pdb,
file_res
character (len=80) ::  c_script
Character (len=30) :: part1
Character (len=24) :: temp, part2
character (len=3) :: c_ver
Character (len=14) :: s_rc, s_k

Real*8, Allocatable :: Uxyz(:,:), GSxyz(:,:), Ori_xyz(:,:)
Real*8 :: DT_F
Real*8 :: Res, Res2
! production run parameters
```

```
Real*8, dimension(2) :: r_range, k_range
Integer :: n_of_r, n_of_k, i_eq
Integer :: I2, J2, IC1, isa, ios, i_bc

Real*8, Allocatable, Dimension(:) :: rc0, k0, a_rc, b_k, value_k
Integer, parameter :: N_actin = 375

print *,' ------------------------------------------------ '
print *,'     Brute Force Simulation of loaded F-actin     '
print *,' ------------------------------------------------ '
print *,'  (c)copyright 2002-2010, IMPACT lab, UT Austin   '
print *,' ------------------------------------------------ '
print *,'  How to operate?                                 '
print *,'  (1) key in the inforamtion through command line '
print *,'  (2) speed process through predefined input file '
print *,'    [example] %./main_run < [inputfile]           '
print *,' ------------------------------------------------ '

write(*,*)'Do you want to go through a series of cutoff and k (brute
force)? (1:yes/0:no)'
Read *, I_prd
Call Check_Range(I_prd, (/0, 1/))
Print *,'How many monomers?'
Read *, N_monomer
Call Check_Range(N_monomer, (/1, 9/))
N = N_actin * N_monomer
Print *, 'What kind of loading?'
Print *,' (1)tension; (2)compresion; (3)bending; (4)Torsion; (5)Shear'
Read*, I_Load
Call Check_Range(I_load, (/1, 5/))

if (I_load == 1) then
  INQUIRE(FILE= '9mer_load0924.inp', EXIST=file_exists)
  if (file_exists == .FALSE.) STOP '[ERROR] load information file
9mer_load0924.inp not exist'
else if (I_load == 4) then
  INQUIRE(FILE= '9mer_torque_load.inp', EXIST=file_exists)
  if (file_exists == .FALSE.) STOP '[ERROR] load information file
9mer_torque_load.inp not exist'
end if

Print *, 'Amount of Load? (pN)'
Read *, DT_F
Print *,'What is the predefine csv file? (*.csv)'
Read *, csv_file
  INQUIRE(FILE= trim(csv_file), EXIST=file_exists)
  if (file_exists == .FALSE.) STOP '[ERROR] cvs file not exist'

Print * ,'What is the SMD (Golden Standard) file name? (*.pdb)'
Read *, GS_file
  INQUIRE(FILE= trim(GS_file), EXIST=file_exists)
  if (file_exists == .FALSE.) STOP '[ERROR] Target SMD file not exist'

print *, 'which version of Calculix? '
```

```
read *, c_ver
print *, 'ProFit: fit the middle monomer or the whole structure? (0:
middle, 1: whole) '
read *, i_fit
Call Check_Range(I_fit, (/0, 1/))
print *, 'BCs in CalculiX: (0: load on 1 end w/ 20 pts pinned, 1: load on
both ends w/ 3 pts pinned) '
print *, '                 (2: load on 4 monomers of 2 ends w/ 3 pts
pinned) -- only for tension load'
read *, i_bc
Call Check_Range(i_bc, (/0, 2/))

if (I_prd == 0) then
   print *, 'what is the cut off radius?'
   Read *, r_c

     if (log10(r_c)>=2.0) STOP 'r_c should be only in 2 digids'
     if (r_c <=3.0) STOP 'r_c too small'

   Print *, 'What is the desired k? '
   Read *, k

     if (k <=0.0) STOP 'k should be a positive number'

   Print *, 'Do you want to see if profit works? (1: yes, others: no)'
   Read*, i_debug
   n_of_r=1
   n_of_k=1
   write(s_rc,'(E14.8)') r_c
   write(s_k,'(E14.8)') k

   file_res = 'res_r'//s_rc//'_k'//s_k//'.txt'
   n_of_k=1
   n_of_r=1
elseif (I_prd==1) then
   print *, 'what is the range of cutoff radius?'
   print *, 'maximum value?'
   read *, r_range(2)
   print *, 'minimum value?'
   read *, r_range(1)

     if (log10(r_range(1))>=2.0) STOP 'r_c should be only in 2 digids'
     if (r_range(1) <=3.0) STOP 'r_c too small'

   print *, 'what is the range of spring constant?'
   print *, 'maximum value?'
   read *, k_range(2)
   print *, 'minimum value?'
   read *, k_range(1)

     if (k_range(1) <=0.0) STOP 'k should be a positive number'

   print *, 'For cutoff radius, how many data points in between?'
   read *, n_of_r
```

```
   print *, 'For spring constant, how many data points in between?'
   read *, n_of_k

   i_debug=0
   print *, 'the output file name: '
   Read *, file_res

   CALL DATE_AND_TIME(BEN1, BEN2,ZONE_T, TIME)
   call random_seed()
   call random_number(K_INDEX)
   K_index=K_index*100D0
   C2_INDEX=ACHAR(INT(MOD(K_INDEX,26D0))+65)

file_res=trim(file_res)//BEN1(6:8)//BEN2(1:2)//BEN2(8:8)//C2_INDEX//'.txt'
   print *,'Circular search on a parabolic equations (0: no; >1: numbers of
parabolic equations)?'
   Read *, I_eq
   if (i_eq>=1) then
      allocate(rc0(i_eq), k0(i_eq), a_rc(i_eq), b_k(i_eq), value_k(i_eq))
      do I=1, i_eq
         print *, 'rc_0? '
         read *, rc0(i)
         print *, 'k_0?? '
         read *, k0(i)
         print *, 'a? '
         read *, a_rc(i)
         print *, 'b? '
         read *, b_k(i)
      end do
   end if
else
   stop 'please choose 0/1'
end if

! ================================= !
!     information of the smd file    !
! ================================= !
open (unit = 22, file = gs_file, status = 'old', iostat = ios)
if (ios /=0) Stop 'open GS file error'

if (allocated(GSxyz)) deallocate(GSxyz)
   allocate(GSxyz(N,3), stat = ios)
if (ios /= 0) stop 'allocation error on GSxyz'

Do I=1,N
   read(22,'(A30,3F8.3, A24)', iostat=ios) part1, GSxyz(I,1:3), part2
   if (ios /= 0) stop 'reading error on loaded SMD file'
End Do
Close(22)

Call res_cal(Gsxyz, N, GS_Res(1:9))


! ================================================== !
```

```
! calculate the undeformed information based on csv file !
! ================================================== !
open (unit = 29, file = csv_file, status = 'old', iostat = ios)
if (ios /=0) Stop 'open CSV file error'

if (allocated(Ori_xyz)) deallocate(Ori_xyz)
   allocate(Ori_xyz(N,3), stat = ios)
if (ios /= 0) stop 'allocation error on Ori_xyz'

Do I=1,N
   read(29,'(6X,F8.3,1X,F8.3,1X,F8.3)', iostat=ios) Ori_xyz(I,1:3)
   if (ios /= 0) stop 'reading error on CSV file'
End Do
Close(29)

Call res_cal(Ori_xyz, N, Ori_Res)

GS_res(10) = Ori_Res(1)/1000.0d0*DT_F/actin_k+Ori_Res(1)

! ================================= !
!  Gold Standard information stored  !
! ================================= !

! ---- Recording Text file --------- !

open  (unit = 50, file = trim(file_res))
if (i_prd == 0) then
   write(50,'(A)') 'This is a test run of Residual calculation'
   write(50,'(A)') '
================================================================================
=== '

   if (i_fit == 0) then
      write(50,'(A)') 'The ProFit zone applied to the middle monomer only'
   else if (i_fit==1) then
      write(50,'(A)') 'The ProFit zone tried to fit the whole set'
   end if

   if (i_set == 0) then
      write(50,'(A)') 'The RMS calculation is based on the whole set '
   else if (i_set==1) then
      write(50,'(A)') 'The RMS calculation is based on the middle monomer'
   end if
   write(50,'(A,A)') 'Calculix Version : ', c_ver
   write(50,'(A,I1)')'Loading condition: (1)tension; (2)compresion;
(3)bending; (4)Torsion; (5)Shear : ',I_load
   write(50,'(A,F10.4)') 'Appling force(torque) : ', DT_F
   write(50,'(A)') '
================================================================================
=== '
   write(50,*) ''
   write(50,'(A,F12.5)') 'r_c = ', r_c
   write(50,'(A,F12.5)') 'k   = ', k
```

```
   write(50,'(34x, 13A16)') ' (1) RMSD all  |','(2)rmsd mid-mer|','(3)
len_total |','(4)   len_m1   |',&
                            ' (5) len m2    |','(6)    B1      |','(7)
B2     |','(8) res288 dis |',&
                            ' (9) angle_134 |','(10) angle_312 |','(11)
Dihedral |','(12) experiment|',&
                            '(13) Dali_Score'
end if

allocate(Res_set(n_of_r*n_of_k,15), stat = isa0)
  if (isa0/=0) stop 'allocation error at Res_set'
Res_set=0.0d0

! ------------------------ !
! Start Main Program       !
! ------------------------ !

IC1=0

Do J2 = 1, n_of_r
   if (i_prd ==1) r_c = r_range(1) + (r_range(2)-r_range(1))/real(n_of_r-
1)*real(J2-1)
   call r_def(r_c, N, csv_file)
    if (allocated(Uxyz)) deallocate(Uxyz)
   Allocate(Uxyz(N,4), stat=isa0)
    if (isa0/=0) Stop 'Allocation Error at Uxyz '
    if (i_eq>=1) then
       do i = 1, i_eq
          value_k(i) = sqrt((1+(r_c-rc0(i))*(r_c-
rc0(i))/a_rc(i)/a_rc(i))*b_k(i)*b_k(i))+k0(i)
       end do
       k_range(1) = minval(value_k)-0.5d0
       k_range(2) = maxval(value_k)+1.0d0
    end if

   Do I2 = 1, n_of_k
      IC1 = IC1+1

      if (i_prd ==1) k=k_range(1) + (k_range(2)-k_range(1))/real(n_of_k-
1)*real(I2-1)
      Res_set(Ic1, 1) = r_c
      Res_set(Ic1 ,2) = k

      Uxyz=0d0

      call femrun(r_c, k, I_load, DT_F, N, Uxyz, N1, csv_file, filename,
c_ver, i_bc)
      call profit_run(GS_file, filename, Uxyz, N, i_fit)

      fit_file=trim(filename)//'fit.pdb'
      irange(1)=1
      irange(2)=N
      call rmsd_pdb(GS_file, fit_file, irange,Res)
      Res_set(IC1,3) = Res
```

```
      irange(1)=((N/375+1)/2-1)*375+1
      irange(2)=((N/375+1)/2)*375
      call rmsd_pdb(GS_file, fit_file, irange,Res)
      Res_set(IC1,4) = Res

      if (i_prd == 0) then
         old_pdb = trim(filename)//'.pdb'
         call rmsd_pdb(GS_file, old_pdb, irange,Res2)
         print *,'Original Residual = ', Res2
      end if

      if (i_prd == 1) then
         ! ------------------ !
         ! clean up the trash !
         ! ------------------ !
         c_script = 'rm '//trim(filename)//'*'
         call system(c_script)
      end if

      Call res_cal(Uxyz(1:N,2:4), N, Res_set(IC1,5:13))
      Res_set(IC1,14) = Res_set(IC1,5)

      Call Dali_score(Uxyz(1:N,2:4), GSxyz, N, Res_set(IC1,15),irange)

      Write(50,'(15E16.7)') Res_set(IC1,1:4), (Res_set(IC1,5)-
GS_Res(1))/GS_Res(1), &
           (Res_set(IC1,6)-GS_Res(2))/GS_Res(2),(Res_set(IC1,7)-
GS_Res(3))/GS_Res(3),&
           (Res_set(IC1,8)-GS_Res(4))/GS_Res(4),(Res_set(IC1,9)-
GS_Res(5))/GS_Res(5),&
           (Res_set(IC1,10)-GS_Res(6))/GS_Res(6),(Res_set(IC1,11)-
GS_Res(7))/GS_Res(7),&
           (Res_set(IC1,12)-GS_Res(8))/GS_Res(8),(Res_set(IC1,13)-
GS_Res(9))/GS_Res(9),&
           (Res_set(IC1,14)-GS_Res(10))/GS_Res(10), Res_Set(IC1,15)
   End Do
   if (allocated(Uxyz)) then
      Deallocate(Uxyz, stat = isa)
      if (isa/=0) stop 'deallocation error at Uxyz'
   end if
End Do


print *,'Residual is saved in: ', file_res

if (i_prd==0) then
   Write(50,'(2E16.7,32x,10E16.7)') Res_set(1,1:2), Res_set(1,5:14)
   write(50,*) ' comparrring to unloaded'
   Write(50,'(2E16.7,32x,9E16.7)') Res_set(1,1:2), Ori_Res(1:9)
   write(50,*) ' comparrring to loaded'
   Write(50,'(2E16.7,32x,10E16.7)') Res_set(1,1:2), GS_Res(1:10)
   write(50,*) ''
```

```
   write(50,'(A)') '   ========================= associate files
================================== '
   write(50,'(A,A)') 'The predefined coordinate file is :  ',csv_file
   write(50,'(A,A)') 'The SMD simulated coordinate file is :  ',gs_file
   write(50,'(A,A4,A1,A5,F6.3,A4)') &
        'The associated connection file of defined r_c is :
','els_',csv_file(1:1),'_conn_',r_c,'.inp'
   write(50,'(A,A)') 'The FEM stretched pdb file is :  ',old_pdb
   write(50,'(A,A)') 'The FEM stretched pdb after ProFit is :  ',fit_file
end if
close(50)

c_script ='cat '//trim(file_res)

 if (i_prd==0) call system(trim(c_script))

STOP
End Program Main_rk
```

Appendix E

CalculiX Axial Loading Input File Code

```
*HEADING
9MER_RC9.49
*NODE,NSET=Ca
*INCLUDE,INPUT=9mer_node.csv
*ELEMENT,TYPE=SPRINGA,ELSET=test
*INCLUDE,INPUT=els_9_conn_9_49.inp
*SPRING,ELSET=test

43.3
*STEP,AMPLITUDE=RAMP
*STATIC
*CLOAD
**mer1,,
1,1,0.02995
1,2,-0.06406
1,3,-0.25712
2,1,0.02995
2,2,-0.06406
2,3,-0.25712
3,1,0.02995
3,2,-0.06406
3,3,-0.25712
.
.
.
372,1,0.02995
372,2,-0.06406
372,3,-0.25712
373,1,0.02995
373,2,-0.06406
373,3,-0.25712
374,1,0.02995
374,2,-0.06406
374,3,-0.25712
**mer2,,
376,1,-0.00827
376,2,-0.04599
376,3,-0.26254
377,1,-0.00827
377,2,-0.04599
377,3,-0.26254
378,1,-0.00827
378,2,-0.04599
378,3,-0.26254
```

```
.
.
.
748,1,-0.00827
748,2,-0.04599
748,3,-0.26254
749,1,-0.00827
749,2,-0.04599
749,3,-0.26254
750,1,-0.00827
750,2,-0.04599
750,3,-0.26254
**mer8,,
2626,1,0.00827
2626,2,0.04599
2626,3,0.26254
2627,1,0.00827
2627,2,0.04599
2627,3,0.26254
2628,1,0.00827
2628,2,0.04599
2628,3,0.26254
.
.
.
2998,1,0.00827
2998,2,0.04599
2998,3,0.26254
2999,1,0.00827
2999,2,0.04599
2999,3,0.26254
3000,1,0.00827
3000,2,0.04599
3000,3,0.26254
**mer9,,
3001,1,-0.02995
3001,2,0.06406
3001,3,0.25712
3002,1,-0.02995
3002,2,0.06406
3002,3,0.25712
3003,1,-0.02995
3003,2,0.06406
3003,3,0.25712
.
.
.
3373,1,-0.02995
3373,2,0.06406
3373,3,0.25712
3374,1,-0.02995
3374,2,0.06406
3374,3,0.25712
3375,1,-0.02995
```

```
3375,2,0.06406
3375,3,0.25712
*BOUNDARY,TYPE=DISPLACEMENT
** Note that the following 3 points are the closest residues to the
** centerline of the vector between monomers 2 & 8
2164,1,3,0.
1188,1,3,0.
1189,1,3,0.
*NODE PRINT,NSET=Ca
U
*EL PRINT,ELSET=test
S,E
*NODE FILE,NSET=Ca
U
*EL FILE,ELSET=test
S,E
*END STEP
```

Appendix F

CalculiX Torsional Loading Input File Code

```
*HEADING
9MER_RC9.49
*NODE,NSET=Ca
*INCLUDE,INPUT=9mer_node.csv
*ELEMENT,TYPE=SPRINGA,ELSET=test
*INCLUDE,INPUT=els_9_conn_9_49.inp
*SPRING,ELSET=test

43.3
*STEP,AMPLITUDE=RAMP
*STATIC
*CLOAD
**mer1,,
1,1,0.20669
1,2,-0.20669
1,3,0.00000
2,1,0.20669
2,2,-0.20669
2,3,0.00000
3,1,0.20669
3,2,-0.20669
3,3,0.00000
.
.
.
373,1,0.20669
373,2,-0.20669
373,3,0.00000
374,1,0.20669
374,2,-0.20669
374,3,0.00000
375,1,0.20669
375,2,-0.20669
375,3,0.00000
**mer2,,
376,1,-0.17711
376,2,0.17711
376,3,0.00000
377,1,-0.17711
377,2,0.17711
377,3,0.00000
378,1,-0.17711
378,2,0.17711
378,3,0.00000
.
```

```
.
.
748,1,-0.17711
748,2,0.17711
748,3,0.00000
749,1,-0.17711
749,2,0.17711
749,3,0.00000
750,1,-0.17711
750,2,0.17711
750,3,0.00000
**mer8,,
2626,1,-0.22140
2626,2,-0.22140
2626,3,0.00000
2627,1,-0.22140
2627,2,-0.22140
2627,3,0.00000
2628,1,-0.22140
2628,2,-0.22140
2628,3,0.00000
.
.
.
2759,1,-0.22140
2759,2,-0.22140
2759,3,0.00000
2760,1,-0.22140
2760,2,-0.22140
2760,3,0.00000
2761,1,-0.22140
2761,2,-0.22140
2761,3,0.00000
**mer9,,
3001,1,0.16380
3001,2,0.16380
3001,3,0.00000
3002,1,0.16380
3002,2,0.16380
3002,3,0.00000
3003,1,0.16380
3003,2,0.16380
3003,3,0.00000
.
.
.
3373,1,0.16380
3373,2,0.16380
3373,3,0.00000
3374,1,0.16380
3374,2,0.16380
3374,3,0.00000
3375,1,0.16380
3375,2,0.16380
```

```
3375,3,0.00000
*BOUNDARY,TYPE=DISPLACEMENT
** Note that the following 3 points are the closest residues to the
** centerline of the vector between monomers 2 & 8
2164,1,3,0.
1188,1,3,0.
1189,1,3,0.
*NODE PRINT,NSET=Ca
U
*EL PRINT,ELSET=test
S,E
*NODE FILE,NSET=Ca
U
*EL FILE,ELSET=test
S,E
*END STEP
```

Bibliography

1. Chhabra, D. and C.G. Remedios, *Actin: An Overview of Its Structure and Function.* Actin-Binding Proteins and Disease, 2008: p. 1-15.
2. Lefranc, F., J. Brotchi, and R. Kiss, *Possible future issues in the treatment of glioblastomas: special emphasis on cell migration and the resistance of migrating glioblastoma cells to apoptosis.* Journal of Clinical Oncology, 2005. **23**(10): p. 2411.
3. van Leeuwen, F.N., et al., *Rac regulates phosphorylation of the myosin-II heavy chain, actinomyosin disassembly and cell spreading.* Nat. Cell Biol, 1999. **1**: p. 242-248.
4. Wakatsuki, T., R.B. Wysolmerski, and E.L. Elson, *Mechanics of cell spreading: role of myosin II.* Journal of Cell Science, 2003. **116**(8): p. 1617-1626.
5. Totsukawa, G., et al., *Distinct roles of MLCK and ROCK in the regulation of membrane protrusions and focal adhesion dynamics during cell migration of fibroblasts.* The Journal of cell biology, 2004. **164**(3): p. 427.
6. Clark, K., et al., *TRPM7, a novel regulator of actomyosin contractility and cell adhesion.* The EMBO Journal, 2006. **25**(2): p. 290-301.
7. Wylie, S.R. and P.D. Chantler, *Separate but linked functions of conventional myosins modulate adhesion and neurite outgrowth.* Nature cell biology, 2000. **3**(1): p. 88-92.
8. Paluch, E., et al., *Dynamic modes of the cortical actomyosin gel during cell locomotion and division.* Trends in cell biology, 2006. **16**(1): p. 5-10.
9. Engler, A.J., et al., *Matrix elasticity directs stem cell lineage specification.* Cell, 2006. **126**(4): p. 677-689.
10. McBeath, R., et al., *Cell shape, cytoskeletal tension, and RhoA regulate stem cell lineage commitment.* Developmental cell, 2004. **6**(4): p. 483-495.
11. Valle, F., M. Sandal, and B. Samor, *The interplay between chemistry and mechanics in the transduction of a mechanical signal into a biochemical function.* Physics of Life Reviews, 2007. **4**(3): p. 157-188.
12. Akst, J., *Full Speed Ahead.* The Scientist, 2009. **23**(12): p. 26-32.
13. Gao, M., et al., *Identifying unfolding intermediates of FN-III10 by steered molecular dynamics.* Journal of molecular biology, 2002. **323**(5): p. 939-950.
14. Grandi, F., et al., *Hierarchical mechanochemical switches in angiostatin.* Chembiochem, 2006. **7**(11): p. 1774-1782.
15. Gittes, F., et al., *Flexural rigidity of microtubules and actin filaments measured from thermal fluctuations in shape.* The Journal of cell biology, 1993. **120**(4): p. 923.

16. Isambert, H., et al., *Flexibility of actin filaments derived from thermal fluctuations. Effect of bound nucleotide, phalloidin, and muscle regulatory proteins.* Journal of Biological Chemistry, 1995. **270**(19): p. 11437.
17. Liu, X. and G.H. Pollack, *Mechanics of F-actin characterized with microfabricated cantilevers.* Biophysical journal, 2002. **83**(5): p. 2705-2715.
18. Nyitrai, M., et al., *The flexibility of actin filaments as revealed by fluorescence resonance energy transfer.* Journal of Biological Chemistry, 1999. **274**(19): p. 12996.
19. Tsuda, Y., et al., *Torsional rigidity of single actin filaments and actin–actin bond breaking force under torsion measured directly by in vitro micromanipulation.* Proceedings of the National Academy of Sciences of the United States of America, 1996. **93**(23): p. 12937.
20. Kojima, H., A. Ishijima, and T. Yanagida, *Direct measurement of stiffness of single actin filaments with and without tropomyosin by in vitro nanomanipulation.* Proceedings of the National Academy of Sciences of the United States of America, 1994. **91**(26): p. 12962.
21. Pfaendtner, J., et al., *Structure and dynamics of the actin filament.* Journal of molecular biology, 2009. **396**(2): p. 252-263.
22. Chu, J.W. and G.A. Voth, *Allostery of actin filaments: molecular dynamics simulations and coarse-grained analysis.* Proceedings of the National Academy of Sciences of the United States of America, 2005. **102**(37): p. 13111.
23. Zheng, X., K. Diraviyam, and D. Sept, *Nucleotide effects on the structure and dynamics of actin.* Biophysical journal, 2007. **93**(4): p. 1277-1283.
24. Tuckerman, M.E., B.J. Berne, and G.J. Martyna, *Molecular dynamics algorithm for multiple time scales: Systems with long range forces.* The Journal of chemical physics, 1991. **94**: p. 6811.
25. Eyal, E. and I. Bahar, *Toward a molecular understanding of the anisotropic response of proteins to external forces: insights from elastic network models.* Biophysical journal, 2008. **94**(9): p. 3424-3435.
26. Engel, J., et al., *The polymerization reaction of muscle actin.* Molecular and Cellular Biochemistry, 1977. **18**(1): p. 3-13.
27. Goldschmidt-Clermont, P.J., et al., *The control of actin nucleotide exchange by thymosin b4 and profilin. A potential regulatory mechanism for actin polymerization in cells.* Mol. Biol. Cell, 1992. **3**: p. 1015–1024.
28. Dalhaimer, P., T.D. Pollard, and B.J. Nolen, *Nucleotide-mediated conformational changes of monomeric actin and Arp3 studied by molecular dynamics simulations.* Journal of molecular biology, 2008. **376**(1): p. 166-183.
29. Schutt, C.E., et al., *The structure of crystalline profilin-ß-actin.* Nature, 1993. **365**: p. 810-816.
30. McLaughlin, P.J., et al., *Structure of gelsolin segment 1-actin complex and the mechanism of filament severing.* 1993.
31. Kabsch, W., et al., *Atomic structure of the actin: DNase I complex.* 1990.

32. Oda, T., et al., *The nature of the globular-to fibrous-actin transition.* Nature, 2009. **457**(7228): p. 441-445.
33. DeLano, W.L., *The PyMOL molecular graphics system*. 2002.
34. Lorenz, M., D. Popp, and K.C. Holmes, *Refinement of the F-actin model against X-ray fiber diffraction data by the use of a directed mutation algorithm.* Journal of molecular biology, 1993. **234**(3): p. 826-836.
35. Otterbein, L.R., P. Graceffa, and R. Dominguez, *The crystal structure of uncomplexed actin in the ADP state.* Science, 2001. **293**(5530): p. 708.
36. Morse, D.C., *Viscoelasticity of concentrated isotropic solutions of semiflexible polymers. 1. Model and stress tensor.* Macromolecules, 1998. **31**(20): p. 7030-7043.
37. Cooper, G.M. and R.E. Hausman, *The cell: a molecular approach*. 2000: ASM Press Washington, DC.
38. Go, N., T. Noguti, and T. Nishikawa, *Dynamics of a small globular protein in terms of low-frequency vibrational modes.* Proceedings of the National Academy of Sciences, 1983. **80**(12): p. 3696.
39. Atilgan, A.R., et al., *Anisotropy of fluctuation dynamics of proteins with an elastic network model.* Biophysical journal, 2001. **80**(1): p. 505-515.
40. Bahar, I., A.R. Atilgan, and B. Erman, *Direct evaluation of thermal fluctuations in proteins using a single-parameter harmonic potential.* Folding and Design, 1997. **2**(3): p. 173-181.
41. Chennubhotla, C., et al., *Elastic network models for understanding biomolecular machinery.* Physical Biology, 2005. **2**: p. S173-S180.
42. Tirion, M.M., *Large amplitude elastic motions in proteins from a single-parameter, atomic analysis.* Physical review letters, 1996. **77**(9): p. 1905-1908.
43. Cui, Q. and I. Bahar, *Normal mode analysis: theory and applications to biological and chemical systems*. 2006: CRC Press.
44. Eyal, E., L.W. Yang, and I. Bahar, *Anisotropic network model: systematic evaluation and a new web interface.* Bioinformatics, 2006. **22**(21): p. 2619.
45. Navidi, W., *Statistics for engineers and scientists*. 2008: McGraw-Hill Higher Education.
46. Debye, P., *Interferenz von Röntgenstrahlen und Wärmebewegung.* Annalen der Physik, 1913. **348**: p. 49-92.
47. Waller, I., *Zur Frage der Einwirkung der Wärmebewegung auf die Interferenz von Röntgenstrahlen.* Zeitschrift für Physik A Hadrons and Nuclei, 1923. **17**(1): p. 398-408.
48. Sweet, T. *One-Dimensional Spring Element*. FEMur, 1-D Spring Element 1999 [cited 2010 October 26]; Available from: http://comp.uark.edu/~jjrencis/femur/Learning-Modules/Stress-Analysis/One-Dimensional-Elements/Spring-Element/spring.htm.
49. Bathe, K.J. and H. Saunders, *Finite element procedures in engineering analysis.* Journal of Pressure Vessel Technology, 1984. **106**: p. 421.

50. Dhondt, G.D.C. and J. Wiley, *The finite element method for three-dimensional thermomechanical applications*. 2004: Wiley Online Library.
51. Van Der Spoel, D., et al., *GROMACS: fast, flexible, and free.* Journal of computational chemistry, 2005. **26**(16): p. 1701-1718.
52. Yasuda, R., H. Miyata, and K. Kinosita Jr, *Direct measurement of the torsional rigidity of single actin filaments.* Journal of molecular biology, 1996. **263**(2): p. 227-236.
53. Chu, J.W. and G.A. Voth, *Coarse-grained modeling of the actin filament derived from atomistic-scale simulations.* Biophysical journal, 2006. **90**(5): p. 1572-1582.
54. Ming, D., et al., *Simulation of F-actin filaments of several microns.* Biophysical journal, 2003. **85**(1): p. 27-35.
55. Noid, W.G., et al., *The multiscale coarse-graining method. I. A rigorous bridge between atomistic and coarse-grained models.* The Journal of chemical physics, 2008. **128**: p. 244114.
56. Sept, D. and F.C. MacKintosh, *Microtubule elasticity: Connecting all-atom simulations with continuum mechanics.* Physical Review Letters. **104**(1): p. 18101.
57. Bathe, M., *A finite element framework for computation of protein normal modes and mechanical response.* Proteins: Structure, Function, and Bioinformatics, 2008. **70**(4): p. 1595-1609.
58. Yang, L.W. and C.P. Chng, *Coarse-grained models reveal functional dynamics-I. elastic network models–theories, comparisons and perspectives.* Bioinformatics and Biology Insights, 2008. **2**: p. 25.
59. Ming, D. and M.E. Wall, *Allostery in a coarse-grained model of protein dynamics.* Physical review letters, 2005. **95**(19): p. 198103.
60. Kondrashov, D.A., et al., *Protein structural variation in computational models and crystallographic data.* Structure, 2007. **15**(2): p. 169-177.
61. Kondrashov, D.A., Q. Cui, and G.N. Phillips Jr, *Optimization and evaluation of a coarse-grained model of protein motion using X-ray crystal data.* Biophysical journal, 2006. **91**(8): p. 2760-2767.
62. Sadd, M.H., *Elasticity: theory, applications, and numerics*. 2009: Academic Press.
63. Atkinson, S.J., M.A. Hosford, and B.A. Molitoris, *Mechanism of actin polymerization in cellular ATP depletion.* Journal of Biological Chemistry, 2004. **279**(7): p. 5194.
64. Avitabile, P., *MODAL SPACE: I made a stiffness change to the tip of a cantilever, beam but l can only shift the frequency so far. What's up?* Experimental Techniques, 2009. **33**(4): p. 9-10.
65. Soheilifard, R., D.E. Makarov, and G.J. Rodin, *Critical evaluation of simple network models of protein dynamics and their comparison with crystallographic B-factors.* Physical Biology, 2008. **5**: p. 026008.

66. Yang, L.W., et al., *oGNM: online computation of structural dynamics using the Gaussian Network Model.* Nucleic acids research, 2006. **34**(suppl 2): p. W24.
67. Plasterer, K. *3D-XYZ-Graph*. InterWriteBackgrounds 2010 [cited 2010 November 08]; Available from: http://www.gtcocalcomp.com/erc/interwritebackgrounds/3D-XYZ-Graph.gif.
68. Wakabayashi, K., et al., *X-ray diffraction evidence for the extensibility of actin and myosin filaments during muscle contraction.* Biophysical journal, 1994. **67**(6): p. 2422-2435.
69. Kreuzer, S., *Personal Communication*. 2010.
70. Finer, J.T., R.M. Simmons, and J.A. Spudich, *Single myosin molecule mechanics: piconewton forces and nanometre steps.* Nature, 1994. **368**(6467): p. 113-119.
71. Holm, L. and C. Sander, *Dictionary of recurrent domains in protein structures.* Proteins: Structure, Function, and Bioinformatics, 1998. **33**(1): p. 88-96.
72. Holm, L. and J. Park, *DaliLite workbench for protein structure comparison.* Bioinformatics, 2000. **16**(6): p. 566.
73. Maiorov, V.N. and G.M. Crippen, *Significance of root-mean-square deviation in comparing three-dimensional structures of globular proteins.* Journal of molecular biology, 1994. **235**(2): p. 625-634.
74. Liu, D., *Personal Communication*. 2010.
75. McLachlan, A.D., *Rapid comparison of protein structures.* Acta Crystallographica Section A, 1982. **38**(6): p. 871-873.
76. Khatiblou, E., *Personal Communication*. 2010.
77. Benavraham, D., *Dynamic and elastic properties of F-actin: a normal-modes analysis.* Biophysical journal, 1995. **68**: p. 1231-1245.
78. Bahar, I. and A.J. Rader, *Coarse-grained normal mode analysis in structural biology.* Current opinion in structural biology, 2005. **15**(5): p. 586-592.
79. Kovács, M., et al., *Load-dependent mechanism of nonmuscle myosin 2.* Proceedings of the National Academy of Sciences, 2007. **104**(24): p. 9994.
80. Nyitrai, M. and M.A. Geeves, *Adenosine diphosphate and strain sensitivity in myosin motors.* Philosophical Transactions of the Royal Society B: Biological Sciences, 2004. **359**(1452): p. 1867.
81. Veigel, C., et al., *Load-dependent kinetics of myosin-V can explain its high processivity.* Nature cell biology, 2005. **7**(9): p. 861-869.

Vita

Joel Marquez was born in El Paso, Tx on April 2, 1984. In 2002, he achieved the rank of Eagle Scout and also graduated from Spring High School in Spring, Tx. He then attended the University of Texas at Austin to earn his Bachelor of Science in Mechanical Engineering, graduating in 2006. After returning from a graduate research assistant position at Los Alamos National Laboratories, he returned to UT Austin to begin his Masters work in the fall of 2007.

E-mail address: jdmarquez21@hotmail.com

This thesis was typed by The Author.